Proceedings

TWENTY-EIGHTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

October 3-5, 1990

Allerton House, Monticello, Illinois
Sponsored by the
Coordinated Science Laboratory and the
Department of Electrical and Computer Engineering of the
University of Illinois at Urbana-Champaign

621.3841
AL5
1990

ENGINEERING LIBRARY
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

PROCEEDINGS

TWENTY-EIGHTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

Donna J. Brown
Kameshwar R. Poolla
Conference Co-Chairs

Conference held
October 3-5, 1990
Allerton House
Monticello, Illinois

Sponsored by
The Coordinated Science Laboratory
and
The Department of Electrical and Computer Engineering
of the
UNIVERSITY OF ILLINOIS
at
Urbana-Champaign

621.3811
Al5
1990

Engin.

FOREWORD

The Twenty-Eighth Annual Allerton Conference on Communication, Control, and Computing was held at Allerton House, Monticello, Illinois on October 3, 4, and 5, 1990. The conference was sponsored by the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. These *Proceedings* contain papers presented at the Conference.

We are very grateful to Professor Leslie Valiant of Harvard University for his plenary lecture "A Bridging Model for Parallel Computation." We are also grateful to Professor M. C. Loui of the University of Illinois (and currently at the National Science Foundation) for organizing the invited session on "Concurrent and Distributed Computation" and to the invited speakers: M. Merritt, J. Misra, G. L. Peterson, N. Santoro, and H. R. Strong. We would like to thank all the session chairs and the authors for their contributions.

We appreciate the support of Professor W. K. Jenkins, Director of the Coordinated Science Laboratory and Professor T. N. Trick, Head of the Department of Electrical and Computer Engineering. We would like to thank our colleagues, Professors K. S. Arun, A. Barron, M. T. Başar, T. Ü. Başar, Y. Bresler, G. Cybenko, B. E. Hajek, T. S. Huang, D. L. Jones, P. V. Kokotovic, P. R. Kumar, H. Lee, C. L. Liu, M. C. Loui, J. V. Medanic, S. Meyn, D. C. Munson, W. R. Perkins, L. Pitt, F. P. Preparata, M. B. Pursley, D. V. Sarwate and M. W. Spong, for their generous assistance in reviewing papers and organizing the papers into sessions. We are also indebted to colleagues elsewhere who so willingly stepped in to assist with reviewing.

We are also grateful to Mr. Steven Williams and Ms. Susan Baxley for handling the business affairs of the Conference; to Mr. Robert Ebeling and Ms. Lilian Beck and their assistants for the printing of large numbers of programs and registration packets; to Mr. Jack Gladin and Mr. Bob McFarlane for their assistance in the photography and drafting; to Mr. Gary Eickman, Ms. Stephanie Stoddard,' and Ms. Jody Beskhart and to all the wonderful staff at Allerton House who so cheerfully accommodated our group; and to Dan Jordan who handles the shipping and sales of these *Proceedings* after the Conference. Most especially we want to thank Ms. Francie Bridges, Ms. Dixie Murphy, Ms. Janet Reese, and Ms. Lila Rhoades for their year-round assistance with the Conference in handling announcements, registration, program preparation, Allerton House arrangements, and the editing of these *Proceedings*. They are the real organizers of this Conference.

Donna J. Brown

Kameshwar R. Poolla

ser.

TABLE OF CONTENTS

FOREWORD i

I-A: NEURAL NETWORKS

Chair: George Cybenko
University of Illinois at Urbana-Champaign

LEARNING CONTROL OF DYNAMICAL SYSTEMS USING NEURAL NETWORKS 1
A. Patrikar and J. Provence

A LEARNING AND FORGETTING ALGORITHM IN ASSOCIATIVE MEMORIES: RESULTS INVOLVING PSEUDO INVERSES 11
G. Yen and A. N. Michel

UNIVERSAL APPROXIMATION USING RADIAL-BASIS-FUNCTION NETWORKS . 21
J. Park and I. W. Sandberg

A CONTROLLED GRADIENT SEARCH FOR THE BACK-PROPAGATION ALGORITHM 31
D. T. Flaherty and A. N. Michel

ANALYSIS AND SYNTHESIS OF A CLASS OF DISCRETE-TIME NEURAL NETWORKS WITH STAIRCASE NONLINEARITIES 41
J. Si and A. N. Michel

A CORRELATION-BASED APPROACH FOR SUPERVISED NEURAL-NET CLASSIFIER 51
R. Y. Li and M. Xu

A MODIFIED HO-KASHYAP ALGORITHM FOR BACK-PROPAGATION TRAINING 60
J.-N. Liaw and R.L. Kashyap

I-B: COMBINATORIAL ALGORITHMS

Chair: Carla Savage
North Carolina State University

OPTIMAL SAMPLING STRATEGIES FOR QUICKSORT 62
C. C. McGeoch and J. D. Tygar

A NEW ALGORITHM FOR GENERATING NECKLACES 72
T.M.Y. Wang and C.D. Savage

STACKS, QUEUES, AND DEQUES WITH ORDER-STATISTIC OPERATIONS.......... 82
D. S. Richards and J. S. Salowe

ON SOME COMBINATORIAL OPTIMIZATION PROBLEMS: BOTTLENECK AND CAPACITY ASSIGNMENT PROBLEMS 92
W. Szpankowski

MINIMIZING THE MAXIMUM WIRE LENGTH IN VLSI GRAPH LAYOUTS 102
D. D. Sherlekar

ROUTING WITH SHORT WIRES AND SMALL CHANNEL WIDTH........................ 112
D. Bhatia

ENERGY CONSUMPTION IN MULTILECTIVE AND BOUNDARY VLSI COMPUTATIONS........................ 122
A. Tyagi

I-C: MULTIPLE-ACCESS COMMUNICATIONS

Chair: John Sadowsky
Purdue University

APPROXIMATING THE MEAN WAITING TIME UNDER A TDMA SCHEDULE...... 132
T. K. Philips and S. S. Panwar

INSTABILITY TIMES OF MULTIACCESS BROADCAST CHANNELS..................... 139
R. S. Maier

CONTENTION RESOLUTION AMONG FOUR SENDERS........................ 149
J. E. Jantzen and M. S. Andersland

FH-MFSK MULTIPLE-ACCESS COMMUNICATIONS FOR THE FACTORY ENVIRONMENTS........................ 159
P. Yegani

I-D: CONTROL OF MECHANICAL SYSTEMS

Chair: Mark Spong
University of Illinois at Urbana-Champaign

ON THE STABILITY OF SINGULARLY PERTURBED LINEAR SYSTEMS WHICH HAVE MARGINLY STABLE BOUNDARY LAYER SYSTEMS........................ 162
M. Corless and D. Da

HYPERSTABLE MODEL-FOLLOWING CONTROL OF AN ELECTROMAGNETIC SUSPENSION........................ 172
B.C. Fabien, R.W. Longman, and F. Freudenstien

MODELING AND CONTROL OF RAPID DEPLOYMENT TRUSS STRUCTURE SYSTEMS........................ 182
M. Boesch, J. X. Xu and Ü. Özgüner

ADAPTIVE CONTROL OF FLEXIBLE JOINT ROBOTS WITH GLOBAL STABILITY 192
F. Mrad and S. Ahmad

COOPERATIVE POSITION CONTROL SYSTEM DESIGN FOR TWO ROBOTIC MANIPULATORS........................ 202
A.C. Hu and P.-Y. Woo

ON THE MEASURE OF REPEATABILITY OF REDUNDANT ROBOTS.................. 204
S. Luo and S. Ahmad

PATH AND TRAJECTORY GENERATION FOR ROBOTS........................ 206
T.L. Tschiesche

I-E: PARALLEL COMPUTATION

Chair: Pravin Vaidya
University of Illinois at Urbana-Champaign

ALGORITHMS AND BOUNDS FOR SHORTEST PATHS AND DIAMETER IN FAULTY HYPERCUBES 216
S.-B. Tien and C. S. Raghavendra

RECONFIGURABLE EMBEDDINGS OF A MULTI-DIMENSIONAL MESH IN A HYPERCUBE 226
T. N. Bui and U. Prabhu

'MOVE AND SMOOTH' ROUTING ALGORITHMS ON MESH-CONNECTED COMPUTERS 236
T. Han and D. F. Stanat

A MODIFIED TREE MACHINE 246
S.-Q. Zheng

DYNAMIC PROGRAMMING ON LINEAR SYSTOLIC ARRAYS 254
J. F. Myoupo

A HEURISTIC OF SCHEDULING PARALLEL TASKS AND ITS ANALYSIS 262
Q. Wang and K. H. Cheng

SCHEDULING PRECEDENCE GRAPHS TO MINIMIZE COMPLETION TIME IN PARTITIONABLE PARALLEL ARCHITECTURES 272
H.-A. Choi and B. Narahari

II-A: GRAPH ALGORITHMS

Chair: Herbert Edelsbrunner
University of Illinois at Urbana-Champaign

AN NC APPROXIMATION ALGORITHM FOR THE MINIMUM DEGREE SPANNING TREE PROBLEM 274
M. Fürer and B. Raghavachari

OPTIMAL PARALLEL ALGORITHMS FOR PROBLEMS MODELLED BY A FAMILY OF INTERVALS 282
S. Olariu, J. L. Schwing, and J.-Y. Zhang

CHORD-FREE PATH PROBLEMS ON INTERVAL GRAPHS 292
G. K. Manacher, S. R. Arikati, and C. P. Rangan

AN NC ALGORITHM FOR THE DOMINATION IN THE PERMUTATION GRAPHS. 294
C. Rhee, S. K. Dhall, and S. Lakshmivarahan

AMELIORATING SEQUENCES OF DNC ALGORITHMS FOR SEPARATORS AND BREADTH-FIRST SEARCH IN PLANAR GRAPHS 296
F. Wan

GRAPH LAYOUT USING QUEUES 305
L. S. Heath, F.T. Leighton, and A. L. Rosenberg

AN EFFICIENT ALGORITHM FOR THE MAXIMUM k-CHAIN PROBLEM 315
R. D. Lou and M. Sarrafzadeh

A HEURISTIC ALGORITHM FOR THE k-CENTER PROBLEM WITH COST AND USAGE WEIGHTS 324
Q. Wang and K. H. Cheng

II-B: SIGNAL PROCESSING

Chair: Douglas Jones
University of Illinois at Urbana-Champaign

MEAN-SQUARE-ERROR ANALYSIS OF ESPRIT UNDER RANDOM SENSOR UNCERTAINTIES 334
V. C. Soon and Y. F. Huang

A GEOMETRIC APPROACH TO MULTICHANNEL, MULTIDIMENSIONAL HARMONIC SIGNAL ANALYSIS IN NONSTATIONARY NOISE 344
A. E. Frazho and P. J. Sherman

ROBUST STABILITY OF TWO-DIMENSIONAL DISCRETE INTERVAL MATRICES 354
P. Bauer

A SPACE OF NON-BANDLIMITED SIGNALS THAT CAN BE RECONSTRUCTED FROM NON-UNIFORMLY SPACED SAMPLES 362
M. Djebeniani and D. Cochran

SIGNAL DESIGN-INTEGRAL, DIFFERENTIAL EQUATIONS, SPECTRAL ESTIMATION, CAPACITY COMPUTATION, BINARY RANDOM PROCESSES-ZERO CROSSING PROBLEM 364
G. R. Murthy

THE APPLICATION OF PROJECTION ONTO CONVEX SETS (POCS) TO MULTIPATH CHANNEL ESTIMATION 374
Z. Kostic, E.L. Titlebaum, and M.I. Sezan

ANALYSIS OF THE MEDIAN LMS ADAPTIVE FILTER 376
P. M. Clarkson and G. A. Williamson

IDENTIFICATION AND ESTIMATION OF MULTICHANNEL NONGAUSSIAN MOVING-AVERAGE PROCESSES USING FOURTH-ORDER CUMULANTS 378
Y. Inouye, L. Tong and R. Liu

II-C: COMMUNICATION SYSTEMS

Chair: John Gubner
University of Wisconsin

HANGUP IN DECISION-DIRECTED DIGITAL CARRIER SYNCHRONIZERS.......... 388
M. P. Fitz and W. C. Lindsey

ON THE LOOP NOISE BANDWIDTH OF SCCL 398
K.-C. Chen and L. D. Davisson

THE STATISTICAL CHARACTERIZATION OF THE ACQUISITION PROCESS IN A FIRST ORDER DIGITAL PHASE-LOCKED LOOP 408
J. M. Cramer and M. P. Fitz

VARIABLE DELAY DECODING OF MULTILEVEL CODES 410
M. G. Mulligan and J. N. Livingston

DIGITAL TRANSMISSION OF AMPLITUDE CODED SIGNALS WITH DIVERSITY. 419
R. J. O'Dea and C. C. Lee

A LARGE DEVIATIONS ANALYSIS OF AVALANCHE PHOTODIODE RECEIVERS 429
K. Ben Letaief, R. Papannareddy, and J.S. Sadowsky

PERFORMANCE BOUNDS FOR COHERENT LIGHTWAVE BFSK WITH HARD-DECISION DECODING 438
Y. E. Dallal and S. Shamai (Shitz)

II-D: DISCRETE EVENT SYSTEMS

Chair: Peter Ramadge
Princeton University

ALTERNATIVE ASSIGNMENT AND SEQUENCING SELECTION IN A DETERMINISTIC DISCRETE-EVENT SYSTEM BY MEANS OF THE MINIMAX ALGEBRA APPROACH 448
M. Aicardi, A. Di Febbraro and R. Minciardi

DYNAMICS OF A SWITCHED N BUFFER SYSTEM 455
C. Chase and P. Ramadge

ON DISCRETE STATE REALIZATIONS FOR CONDITION/EVENT SYSTEMS 465
R. S. Sreenivas and B. H. Krogh

RELATIONSHIPS BETWEEN EVENT RATES AND AGGREGATION IN HIERARCHICAL DISCRETE EVENT SYSTEMS 475
K. M. Passino and P. J. Antsaklis

A QUALITATIVE APPROACH TO COMPARING PERFORMANCE OF DISCRETE EVENT SYSTEMS 485
F. Lin

STABILITY ANALYSIS OF DISCRETE EVENT SYSTEMS 487
K. M. Passino, A. N. Michel, and P. J. Antsaklis

II-E: LINEAR SYSTEMS

Chair: Krishan Nagpal
University of Illinois at Urbana-Champaign

ON THE THEORY OF POLYNOMIAL MATRIX INTERPOLATION AND ITS ROLE IN SYSTEMS AND CONTROL 497
P. J. Antsaklis and Z. Gao

CANONICAL FORMS FOR LINEAR TIME-VARYING MULTIVARIABLE DISCRETE SYSTEMS 507
B. P. Park and E. I. Verriest

BLOCKING ZEROS AND DYNAMIC FEEDBACK 517
M. K. Sain and C. B. Schrader

COMPUTATIONAL SIMPLIFICATION IN CONTROLLABILITY AND OBSERVABILITY TESTS 527
S. Bingulac and D. W. Luse

COMPUTATIONAL SIMPLIFICATION IN ERA (EIGENSYSTEM REALIZATION ALGORITHM) 529
B. M. Gorti, S. Bingulac, and H. F. VanLandingham

III-A: CONCURRENT AND DISTRIBUTED COMPUTATION

Chair and Organizer: Michael Loui
Univ. of Illinois/National Science Foundation

†A UNIFIED THEORY OF DISTRIBUTED COORDINATION WITH COMMUNICATION UNCERTAINTY 531
H. R. Strong, D. Dolev, and F. Cristian

†LOOSELY-COUPLED PROCESSES 541
J. Misra

†HIERARCHICAL LAZY CACHING 548
G. Brown and M. Merritt

†COMPUTING WITH TIME (TEMPORAL DIMENSIONS IN DISTRIBUTED COMPUTING) 558
N. Santoro

†OBSERVATIONS ON *l*-EXCLUSION 568
G. L. Peterson

EFFICIENCY OF SEMI-SYNCHRONOUS VERSUS ASYNCHRONOUS NETWORKS. 578
H. Attiya and M. Mavronicolas

CLOCK SYNCHRONIZATION AND THE POWER OF BROADCASTING 588
J. Y. Halpern and I. Suzuki

III-B: IMAGE PROCESSING

Chair: Thomas Huang
University of Illinois at Urbana-Champaign

A NEW ALGORITHM FOR THE RECOVERY OF IMAGES FROM THEIR TRIPLE-CORRELATION FUNCTIONS 598
A. Hammoud, T. J. Schulz, and D. L. Snyder

---

† invited paper

IMAGE RESTORATION ALGORITHMS USING MODIFIED QUADRATIC DISCRIMINANT FUNCTIONS........ 599
S.-J. Ko, T. M. Forest and F. Kimura

LEAST SQUARES RESTORATION OF MULTI-CHANNEL IMAGES 601
N. P. Galatsanos, A. K. Katsaggelos, R. T. Chin, and A. Hillery

TEXTURE SEGMENTATION BASED ON RANDOM FIELD MODELS AND KNOWLEDGE REASONING 603
H. Li and R. L. Kashyap

ROTATION INVARIANT OBJECT DETECTION USING ANOVA-BASED DATA MODEL........ 605
M. H. Benteftifa and L. Kurz

A NEW QNF FOR PYRAMID IMAGE CODING........ 615
C. H. Lu and Z. Lin

ADAPTIVE EDGE DETECTION BASED NESTED MEDIAN FILTER........ 617
H. Hwang and R. A. Haddad

SOME PROPERTIES OF MAX/MIN FILTERS........ 628
C. H. Lu and R. E. Kamel

III-C: QUEUEING SYSTEMS

Chair: Linn Sennott
Univ. of Illinois/Illinois State Univ.

ON THE CONSERVATION LAW AND THE PERFORMANCE SPACE OF SINGLE SERVER SYSTEMS........ 630
L. Georgiadis and I. Viniotis

OPTIMAL ROUTING IN A SIMPLE NETWORK WITH DETERMINISTIC ARRIVAL AND SERVICE TIMES........ 640
M. Aicardi, R. Minciardi, and R. Pesenti

A FAST STOCHASTIC APPROXIMATION ALGORITHM FOR OPTIMIZATION OF DYNAMICAL SYSTEMS........ 650
H.-J. Chang and K. Liano

ON A STOCHASTIC OPTIMIZATION ALGORITHM USING IPA WHICH UPDATES AFTER EVERY CUSTOMER........ 658
E.K.P. Chong and P. J. Ramadge

EXTENSIONS OF PERTURBATION ANALYSIS TO STATIONARY QUEUES........ 668
P. Konstantopoulos and M. Zazanis

PRIORITY LOAD SHARING: AN APPROACH USING STACKELBERG GAMES..... 674
A. A. Economides and J. A. Silvester

III-D: INFORMATION THEORY AND SOURCE CODING

Chair: Andrew Barron
University of Illinois at Urbana-Champaign

RANDOM CODING FOR THE CONSTRAINED MULTIPLE-ACCESS ARBITRARILY VARYING CHANNEL 684
J. A. Gubner

THE ASYMPTOTIC CAPACITY OF THE DIRECT DETECTION PHOTON CHANNEL WITH A BANDWIDTH CONSTRAINT 691
D. Brady and S. Verdu´

RESTRICTED TWO-WAY CHANNELS WITH CORRELATED SOURCES 701
M. Salehi

EXACT ASYMPTOTIC STATISTICS FOR SIGMA-DELTA QUANTIZATION NOISE 703
D. F. Delchamps

OPTIMAL QUANTIZERS: COMPUTATIONAL IMPROVEMENT AND PERFORMANCE 713
S. C. Greer and J. G. Dunham

III-E: CONTROL SYSTEMS

Chair: Andy Packard
University of California, Berkeley

REGULARITY OF THE ZERO DYNAMICS ALGORITHM AND APPROXIMATE INPUT OUTPUT DECOUPLING 715
M. Jankovic and B. K. Ghosh

MINIMAX STATE ESTIMATION FOR SOME NONLINEAR STOCHASTIC SYSTEMS WITH UNCERTAIN STATISTICS 725
E. Yaz

A FRANK-WOLFE/GRADIENT PROJECTION METHOD FOR SOLVING OPTIMAL CONTROL PROBLEMS 733
G.G.L. Meyer and L. J. Podrazik

CONTROLLED INVARIANCE FOR SINGULAR DISTRIBUTIONS 743
V. Ramakrishna

IV-A: ROBUST CONTROL

Chair: Juraj Medanic
University of Illinois at Urbana-Champaign

EXACT COMPUTATION OF THE INFIMUM IN H^{∞}-OPTIMIZATION VIA STATE FEEDBACK 745
B. M. Chen, A. Saberi, and U.-L. Ly

ROBUST DESIGN FOR ILL-CONDITIONED PLANTS WITH DIAGONAL INPUT UNCERTAINTY 755
J. Chen and J. S. Freudenberg

ROBUST STABILITY WITH TIME-VARYING PERTURBATIONS 765
A. Packard and J. Teng

STATE-FEEDBACK SOLUTIONS TO QUADRATIC STABILIZATION PROBLEMS .. 768
A. Packard and G. Becker

COMPLETE INSTABILITY OF A BOX OF POLYNOMIALS 770
P. Padmanabhan and C. V. Hollot

ON H_∞ OPTIMAL STRUCTURAL VARIATIONS 779
D. Bensoussan and H. Maher

CHARACTERISTICS OF RICCATI SOLUTIONS IN A GENERAL H_∞ CONTROL PROBLEM 781
P. Pandey, A. Packard, and A. J. Laub

SCHUR-STABILITY OF A CLASS OF PARAMETER-DEPENDENT MATRICES 783
F. Garofalo, L. Glielmo, and L. Bevilacqua

IV-B: COMPUTATIONAL GEOMETRY

Chair: Franco Preparata
University of Illinois at Urbana-Champaign

MINIMUM WEIGHT QUADRILATERALIZATION IN $O(n^3 \log n)$ TIME 788
H. E. Conn and J. O'Rourke

AN OPTIMAL PARALLEL ALGORITHM FOR THE RECTILINEAR VORONOI DIAGRAM 798
S. Guha

THE POLYGON SEARCH PROBLEM 808
D. Crass, I. Suzuki, and M. Yamashita

EFFICIENT GEOMETRIC ALGORITHMS IN THE EREW-PRAM 818
D. Z. Chen

1-D COMPACTION IN THE PRESENCE OF FORBIDDEN REGIONS 828
S. Hambrusch and H.-Y. Tu

COVERING A SET OF POINTS WITH FIXED SIZE HYPERSQUARES AND RELATED PROBLEMS 838
T. F. Gonzalez

IV-C: COMMUNICATION NETWORKS

Chair: Michael Zazanis
Northwestern University

A SIMPLE STRATEGY FOR DEADLOCK PREVENTION IN STORE-AND-FORWARD NETWORKS 848
S. W. Lau and F.C.M. Lau

A NEW MODEL AND FAST ELECTION IN FIBER-OPTIC RINGS 858
H. Abu-Amara and V. Gummadi

ADAPTIVE ROUTING FOR CONGESTION CONTROL IN MULTIPATH REGULAR NETWORKS 860
R. Krishnan and J. A. Silvester

PERFORMANCE LIMITS FOR FDMA CELLULAR TELEPHONE SYSTEMS 869
R. J. McEliece and K. N. Sivarajan

NEURAL NETWORK IMPLEMENTATION OF THE SHORTEST PATH ALGORITHM FOR TRAFFIC ROUTING IN COMMUNICATION NETWORKS 881
S.C.A. Thomopoulos, L. Zhang, and C. D. Wann

ERGODICITY OF A QUEUEING NETWORK AND AN ALTERNATIVE PROOF OF THE MAXFLOW-MINCUT THEOREM 891
L. Tassiulas and A. Ephremides

AN EFFICIENT DISTRIBUTED ALGORITHM FOR ROUTING ON K-DISJOINT PATHS OF MINIMAL TOTAL LENGTH 901
C. Cheng, S.P.R. Kumar, and J. J. Garcia-Luna

MAXIMIZING CONGESTION-FREE MESSAGE PATTERNS FOR BUNDLED OMEGA NETWORKS 911
P. J. Bernhard and D. J. Rosenkrantz

IV-D: DATA STRUCTURES AND FOUNDATIONS

Chair: David Muller
University of Illinois at Urbana-Champaign

AN IMPROVED DEFERRED DATA STRUCTURE FOR MEMBERSHIP QUERIES.... 914
A. Bhansali and S. S. Skiena

TYPICAL BEHAVIOR OF PATRICIA TRIES 924
B. Rais, P. Jacquet, and W. Szpankowski

PARALLEL PRIORITY QUEUES 926
M. C. Pinotti and G. Pucci

AN OPTIMAL PARALLEL CONSTRUCTION SCHEME FOR HEAP-LIKE STRUCTURES 936
S. Olariu and Z. Wen

PARALLEL DEMAND-DRIVEN EVALUATION OF LOGIC NETWORKS 938
P. E. Dunne, C.J.J. Gittings, and P. H. Leng

THE COSTTABLE PROBLEM IS NP-COMPLETE 948
K. A. Schueller and J. T. Butler

ON FINDING OBSTRUCTIONS FOR LAYOUT PERMUTATION PROBLEMS 958
N. G. Kinnersley

V-A: RELIABLE DISTRIBUTED COMPUTING

Chair: Nancy G. Kinnersley
University of Kansas

AN EFFICIENT ADAPTIVE ALGORITHM FOR DISTRIBUTED SELF DIAGNOSIS IN MULTIPROCESSING SYSTEMS 968
Y. Rosen and R. Gnacek

RELIABLE GOSSIP SCHEMES WITH RANDOM LINK FAILURES 978
K. Diks and A. Pelc

SELF-STABILIZING DISTRIBUTED SYSTEMS WITH BINARY MACHINES 988
S. Ghosh

EFFICIENT, DYNAMICALLY STRUCTURED MULTIPROCESS COMMUNICATION 998
Y.-J. Joung and S. A. Smolka

V-B: CODING FOR MULTIPLE ACCESS COMMUNICATIONS

Chair: Robert McEliece
California Institute of Technology

CASCADED GMW SEQUENCES 1008
A. Klapper, A. H. Chan, and M. Goresky

AN N-USER MULTIPLE ACCESS SYSTEM BASED ON (N,1,m) CONVOLUTIONAL CODES 1014
V. Telang and M. A. Herro

A NEW FAMILY OF ALGEBRAICALLY DESIGNED OPTICAL ORTHOGONAL CODES 1024
Z. Kostic, S. V. Maric, and E. L. Titlebaum

PN-CHIRP SIGNALS FOR MULTIPLE-USER ANTI-MULTIPATH COMMUNICATION 1026
C. H. Suhartanto and C. D. McGillem

V-C: ADAPTIVE CONTROL

Chair: Sean Meyn
University of Illinois at Urbana-Champaign

A MODIFIED OUTPUT ERROR METHOD FOR MODEL REFERENCE ADAPTIVE CONTROL 1028
Y.N.M. Papadakis and S.C.A. Thomopoulos

STATE SPACE APPROACH TO NONLINEAR MODEL REFERENCE ADAPTIVE CONTROL 1038
A.S.C. Sinha, S. Kayalar, and H. O. Yurtseven

FUZZY GAIN SCHEDULING CONTROL OF A NON-LINEAR HVAC SYSTEM........ 1044
G. Geng and A. L. Dexter

V-D: ESTIMATION, DETECTION and RECONSTRUCTION

Chair: David Munson
University of Illinois at Urbana-Champaign

ON THE TIGHTNESS OF CONSTRAINED MULTI-DIMENSIONAL CRAMER-RAO BOUNDS 1053
J. D. Gorman and A. O. Hero

ON THE RELATION OF CROSS-VALIDATION AND ALTERNATIVE CRITERIA TO ESTIMATE THE REGULARIZATION PARAMETER 1061
N. P. Galatsanos and A. K. Katsaggelos

ESTIMATION OF NONGAUSSIAN SIGNALS USING A MODIFIED GAUSSIAN SUM ADAPTIVE FILTER 1063
M. J. Caputi, R. L. Moose and W. T. Baumann

PERFORMANCE OF AN ADAPTIVE DETECTION SCHEME IN AN IMPULSIVE CORRELATED ENVIRONMENT 1072
P. Srivastava and P. Willett

A SELECTION AND ESTIMATION TEST FOR MULTIPLE TARGETS IN CLUTTER DETECTION 1074
R. Viswanathan and A. Eftekhari

INDEX OF AUTHORS 1084

Learning Control of Dynamical Systems using Neural Networks

Ajay Patrikar and John Provence

Department of Electrical Engineering
Southern Methodist University
Dallas, Texas 75275

Abstract

Our objective in this paper is to show that layered neural networks with feedback connections can be used to control plants with unknown system dynamics. During the learning phase the neural network controller improves its performance by repeated trials with the system for a specified reference input. The function of neural network controller is to map states of the system into corresponding system-inputs in order to make the system follow a certain output trajectory which is specified using a reference model. The neural network learns this mapping through error back-propagation as the control process evolves. The application of such control scheme for linear systems and a class of nonlinear systems is shown through simulation.

I. Introduction

Theory of adaptive control developed over the past two decades has found many applications in industry. The present adaptive methods are very successful in controlling linear control systems. However, their application to nonlinear control problems has met with only a limited success. The major obstacle being the lack of tools to model nonlinear systems accurately.

Recently the application of neural networks in adaptive and learning control of dynamical systems has attracted much attention [1]-[3]. The ability of a neural network to achieve arbitrary mapping, if trained sufficiently, has been made use of in many applications. Narendra and Parthasarathy [1] proposed use of neural network for system identification of nonlinear systems and used this neural network system model in model reference adaptive control of dynamical systems. The control scheme proposed in [1] falls under the category of indirect adaptive control [4] because the control parameters are not directly adjusted based on error between the plant output and the reference model output.

Our aim in this paper is to explore the use of neural networks for direct adaptive control of dynamical systems. We show that the parameters (i.e., connection strengths) of the neural network can be directly adjusted based on the output error for a class of dynamical systems, thus avoiding the need for system identification. The control scheme proposed in this paper is termed as a learning control scheme because the neural network controller requires repeated trials with the unknown system to reach satisfactory performance.

The organization of the paper is as follows. The multilayer neural networks used to design the controllers are described in section II. Section III describes the direct adaptive control approach using neural networks and discusses different features of neural network controller. The simulation results on linear and nonlinear systems are presented in Section IV.

II. Multilayer Neural Networks

A multilayer neural network with one hidden layer is shown in Fig. 2 (a). The network consists of linear and nonlinear neurons. The function of each neuron is to take weighted sum of all its inputs and map the sum using a linear or nonlinear mapping as shown in Fig. 2 (b). Let the connection strength between the i^{th} and the j^{th} neuron be denoted by w_{ij}. Then the net input to the j^{th} neuron is given by

$$net_j = \sum_i w_{ij}\, x_i + \theta_j .$$

where x_i is output of the i^{th} neuron and θ_j is the threshold bias associated with neuron j. The bias θ_j can be viewed as a strength of the connection from a neuron whose output is always 1.0 to neuron j. The output x_j of neuron j is then given by

$$x_j = f_j(\, net_j\,) = \frac{1}{1 + e^{-net_j}}$$

if the neuron is nonlinear. For linear neuron the output $x_j = f_j(\, net_j\,) = net_j$.

These kind of neural networks can be used to achieve arbitrary mappings between a set of input and output patterns, if trained sufficiently. In supervised training of neural networks we are given a set of P input patterns X_p, $p = 1,..,P$ which need to be mapped into the corresponding set of desired output patterns $D_p = [d_{1p}, d_{2p}, ..., d_{Mp}]$. Let $Y_p = [y_{1p}, y_{2p}, ..., y_{Mp}]$ be the actual output of the neural network for the p^{th} pattern. To achieve the desired mapping the mean squared error between the actual outputs of the network and the desired outputs i.e.,

$$MSE = \sum_{p=1}^{P} \sum_{m=1}^{M} (\, d_{mp} - y_{mp}\,)^2$$

needs to be minimized. Backpropagation algorithm which is an iterative gradient descent algorithm to minimize the mean squared error can be used for this purpose. In this algorithm the connection strengths and the thresholds of the network are changed at each sample presentation in the following manner:

$$\Delta w_{ij}(t) = -\,\eta\, \delta_j\, x_i + \alpha\, \Delta\, w_{ij}(t-1)$$

where η and α are the constants which determine learning rate and the effect of momentum term respectively. δ_j is the error term for the j^{th} neuron. If the neuron j is output neuron then δ_j is calculated by

$$\delta_j = f'_j(net_j)\ (d_j - y_j)$$

where d_j is the desired output of neuron j, y_j is the actual output and $f'_j(net_j) = df\ (net_j)/d\ net_j$. If the neuron j belongs to the hidden layer then δ_j is calculated as

$$\delta_j = f'_j(net_j) \sum_k \delta_k\ w_{jk}$$

where k is over all the neurons which are connected to neuron j from above.

III. Direct Adaptive Control using Neural Networks

The block diagram of model reference adaptive system (MRAS) using neural network is shown in Fig. 1. The desired response of the unknown plant for a given reference input is specified in terms of a model. The neural network controller is required to generate a signal which would drive the unknown plant to follow the reference model. In direct adaptive control the error between output of the plant and the model output is used to change the parameters of the neural network controller. The direct application of gradient descent algorithm such as backpropagation would require that the output error is propagated backwards through the plant. This would lead to difficulty because system dynamics of the plant is unknown.

A class of dynamical systems for which neural network learning controller is proposed can be described by the following difference equation.

$$y(k+1) = \sum_{i=0}^{n} a_i\ u(k-i) + g(\ y(k), y(k-1), \cdots ,y(k-m)\ )$$

where u(k) and y(k) represent system input and output respectively at k^{th} time instant. a_i, (i= 1,..,n) are the parameters of the system which are assumed to be constant but unknown. The function g is an unknown linear or nonlinear function. This class of dynamical systems consists of all the linear systems and nonlinear systems which depend linearly on the input.

Let the reference model be described by the linear difference equation

$$y_m(k+1) = \sum_{i=0}^{p} \alpha_i\ u_c(k-1) + \sum_{i=0}^{q} \beta_i\ y_m(k-i)$$

where $u_c(k)$ is a given reference signal and $y_m(k)$ is an output of the reference model. Let the output error be defined as $e_c(k) = y(k) - y_m(k)$. The inputs to the controller are the reference input and the past input and output values of the plant which represent state of the system. The controller is required to generate plant input u(k) such that the output error $e_c(k)$ asymptotically approaches zero. Thus the function of the neural network is to map states of the system into appropriate control actions.

The controller and the plant together can be viewed as a feedforward layered network with feedback connections. An example of this is shown in Fig. 3. The network is assumed to be consisting of nonlinear as well as linear neurons. The plant parameters are viewed as connections with fixed but unknown strengths and they are represented by dashed lines in Fig. 3. The input to the network is $\phi(k)$ which is a vector of past input and output values of the system and reference model, i.e.,

$$\phi(k)=[u(k-1),\ldots,u(k-n),y(k-1),\ldots,y(k-r),u_c(k),\ldots,u_c(k-s)].$$

The network takes form of a discrete time nonlinear system. The mapping achieved by the neural network at k^{th} time instant can be written as

$$y(k+1) = h(\ \phi(k),\ \mathbf{w}(k)).$$

The training of neural network involves finding the appropriate set of connection strengths **w** such that output of the neural network follows the desired output trajectory $y_m(k)$, $k > 0$, starting from some specified initial conditions. This implies that the performance criterion $\mathbb{P}$ given by

$$\mathbb{P} = \sum_k (\ y_m(k) - y(k)\ )^2$$

be minimized. This is a supervised learning problem as the desired output is known at each time instant. However for unknown systems some of the parameters or connection strengths which are needed to propagate the error backwards are unknown.

At each sampling instant, the vector $\phi(k)$ is modified and presented at the input of neural network controller. The output of controller u(k) is evaluated and applied at the input of the plant. The error between the plant output and model output is then propagated backwards to adjust the connection strengths of controller. Thus δ_j for the neuron corresponding to the output of controller (i.e., u(k)) is calculated as

$$\delta_j = a_0\ f'_j(net_j)\ (\ y_m(k+1) - y(k+1)\ )$$

This procedure requires that the system parameter a_0 be known. But since a_0 can be absorbed with the learning rate η when the weights are modified, only its sign needs to be known.

The learning phase of the controller consists of several trials. Each trial involves training the neural network for some prespecified amount of time. The plant is then reset and the procedure is repeated starting from the same initial conditions. The controller improves its performance with each trial. This learning procedure differs from static backpropagation in the sense that inputs to the controller are different in each trial as the plant follows a different trajectory in each trial during the learning phase. The controller thus acts like a recurrent network and learns to generate a trajectory of system-inputs to drive the plant to follow the reference model.

The choice of parameters η and α control the learning rate of the controller. High values of these parameters lead to large perturbations in connection strengths at each sampling instants leading to instability. Very small values of these parameters lead to slower learning. The selection of these parameters needs to be made on trial and error basis. Apart from these parameters the system designer has to select elements of the vector $\phi(k)$ in such a way that it represents state of the system adequately. If there are any redundant elements in $\phi(k)$ it would lead to bigger networks which take longer time to learn. A priori knowledge of order of the system would be helpful in making this choice.

The major limitation of the above approach is that the controller gets trained to control the plant for a specified input only, and the same controller may not work for arbitrary input. However in practice where the system is required to perform repetitive tasks, this approach may be useful. For dynamical systems in which the output does not depend linearly on the inputs, methods for propagating error through the plant are not currently available and the indirect adaptive control approach [1] may be used.

IV. Simulation Results

The feasibility of above control scheme was verified through simulation. Some of the results are presented here.

Example I. Consider a plant given by the difference equation

$$y(k+1) = 2.4\ u(k) - 1.92\ u(k-1) + 0.1\ y(k) + 0.42\ y(k-1)$$

which is required to follow a model given by the difference equation

$$y_m(k+1) = 0.25\ u_c(k-1) + y_m(k-1) - 0.25\ y_m(k-1)$$

The dynamics of the plant is assumed to be unknown. Only the sign of coefficient of u(k) is assumed to be known. The reference input is a square wave with magnitude varying between 0 and 1. A neural network with 21 neurons is chosen to implement the controller. There are 4 input neurons, 16 hidden neurons and 1 output neuron. The initial weights of the network were chosen to be random numbers in the interval [-0.1,0.1]. The input vector $\phi(k)$ is defined as

$$\phi(k) = [u(k-1), y(k-1), y(k-2), u_c(k)\ ]$$

The neural network was trained with learning rate $\eta = 0.05$ and $\alpha = 0.5$. Each trial involved training the network for 50 samples starting from zero initial conditions. The learning was measured in terms of a performance index defined as cumulative square of the output error (i.e., cse $= \sum_{k=0}^{50} (y(k) - y_m(k))^2$). The learning curve for this problem is shown in Fig. 4 (a). It can be seen that the performance index reduces with each trial. The performance of controller after 300 trials is shown in Fig. 4 (b). It can be seen that after 300 trials the plant is following the model very closely.

The following two examples deal with nonlinear systems. These problems were attempted using indirect adaptive control methods in ref. [1]. We present the same problems in order to compare the results with those obtained in ref. [1].

Example II. Consider the plant given by the following difference equation

$$y(k+1) = g[y(k), y(k-1)] + u(k)$$

where the function g is given by

$$g[y(k), y(k-1)] = \frac{y(k)\, y(k-1)\, [y(k) + 2.5]}{1 + y^2(k) + y^2(k-1)}.$$

A reference model is described by the difference equation

$$y_m(k+1) = 0.6\, y_m(k) + 0.2\, y_m(k-1) + u_c(k).$$

The reference input is a signal given by $u_c(k) = \sin(2\pi k/25) + \sin(2\pi k/10)$.

A neural network with 21 neurons was chosen to implement the controller. It has 4 input neurons, 1 output neuron and 16 neurons in the hidden layer. The input to controller is $\phi(k) = [y(k-1), y(k-2), u_c(k), u_c(k-1)]$. Each trial involved training the network for 80 samples. The performance criterion for this problem is same as previous example. The controller was trained using $\eta = 0.1$ and $\alpha = 0.4$. Fig. 5 (a) shows the learning curve for this problem. Fig. 5 (b) shows the response of the plant after 500 trials. It closely matches the response of the reference model.

Example III. Consider the plant given by the following difference equation

$$y(k+1) = g[y(k), y(k-1), y(k-2)] + u(k) + 0.8\, u(k-1)$$

where the function g is given by

$$g[y(k), y(k-1), y(k-2)] = \frac{5\, y(k)\, y(k-1)}{1 + y^2(k) + y^2(k-1) + y^2(k-2)}$$

A reference model is described by the difference equation

$$y_m(k+1) = 0.32\, y_m(k) + 0.64\, y_m(k-1) - 0.5 y_m(k-2) + u_c(k).$$

The reference input is a sinusoidal signal given by $u_c(k) = \sin(2\pi k/25)$.

A neural network with 21 neurons was chosen to implement the controller. It has 5 input neurons, 1 output neuron and 15 neurons in the hidden layer. The input to controller is $\phi(k) = [y(k-1), y(k-2), u(k-1), u_c(k), u_c(k-1)]$. The performance criterion for this problem is same as previous example. The controller was trained using $\eta = 0.1$ and $\alpha = 0.4$. Fig. 6 (a) shows the learning curve for this problem. Fig. 6 (b) shows the response of the plant after 400 trials. It closely matches the response of the reference model.

In all the above examples the controllers were trained within 15000 to 40000 samples. Thus learning in these controllers was much faster than the indirect adaptive controllers proposed in ref. [1]. Also the size of networks was much smaller. However the above controllers were trained for specific inputs only.

IV. Conclusion

Most of the currently available adaptive control methods work well for linear systems but face difficulty when applied to nonlinear systems because the controller is often linear in parameters. In this paper we explored the ability of multilayer neural networks to form nonlinear mappings to design controllers for a class of dynamical systems. The control methods were aimed at the nonlinear systems with unknown dynamics. In the adaptive methods proposed, the parameters of the neural network controller were varied directly based on the output error. It is shown that the neural networks learn to control the systems by repeated trials. The controllers proposed in this paper may be useful in situations where the system is required to perform repetitive tasks.

References

[1] K. S. Narendra and K. Parthsarathy. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, vol. 1, No. 1, March 1990.

[2] A. G. Barto, R. S. Sutton and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cyber.* , SMC-13, 834-846, 1983.

[3] C. W. Anderson, Learning to control an inverted pendulum using neural networks. *IEEE Control System Magazine* , vol. 15, No. 3, April 1989.

[4] K. J. Astrom and B. Wittenmark. Adaptive Control. Addison Wesley, 1989.

[5] D. E. Rumelhart, G. E. Hinton and R. J. Williams. "Learning internal representations by error propagation" in D. E. Rumelhart and J. L. McClelland (Eds.), Parallel Distributed Processing, vol. 1 : Foundations. MIT Press (1986).

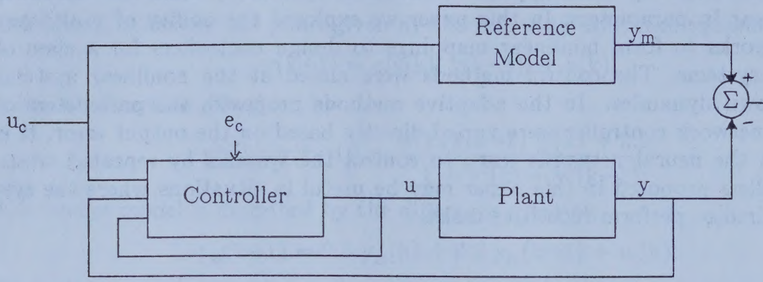
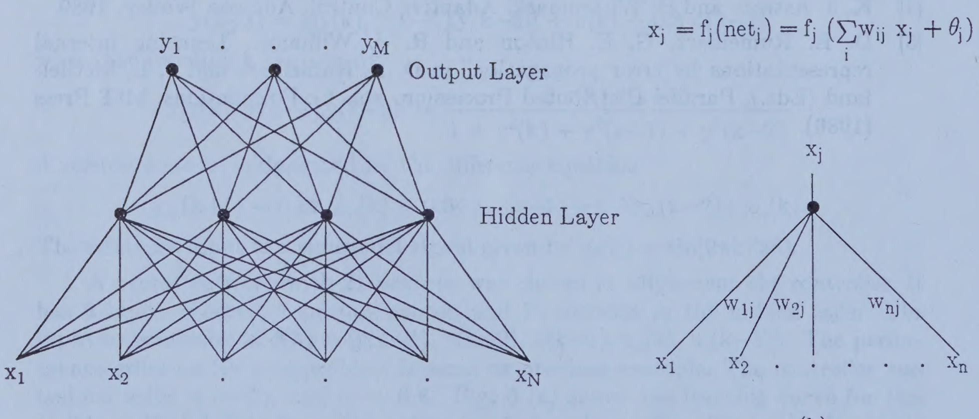

Reference Model
y_m
+
Σ
e_c
−
u_c
e_c
Controller
u
Plant
y

Fig. 1. Direct adaptive control using neural network controller.


y_1 . . . y_M
Output Layer
Hidden Layer
x_1 x_2 . . . x_N
x_j
w_{1j} w_{2j} w_{nj}
x_1 x_2 . . . . x_n

$$x_j = f_j(\text{net}_j) = f_j\ (\sum_i w_{ij}\ x_j + \theta_j)$$

(a) (b)

Fig. 2. (a) Multilayer feedforward neural network, (b) Function of a neuron.

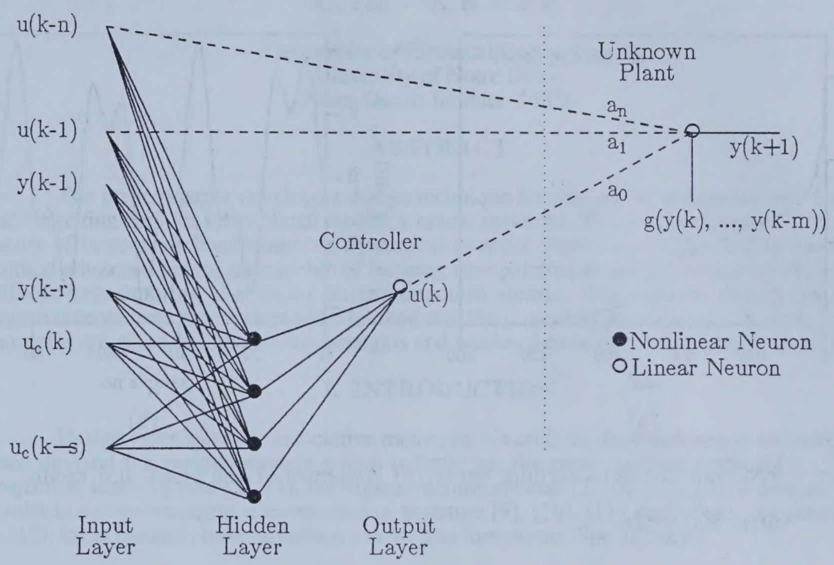
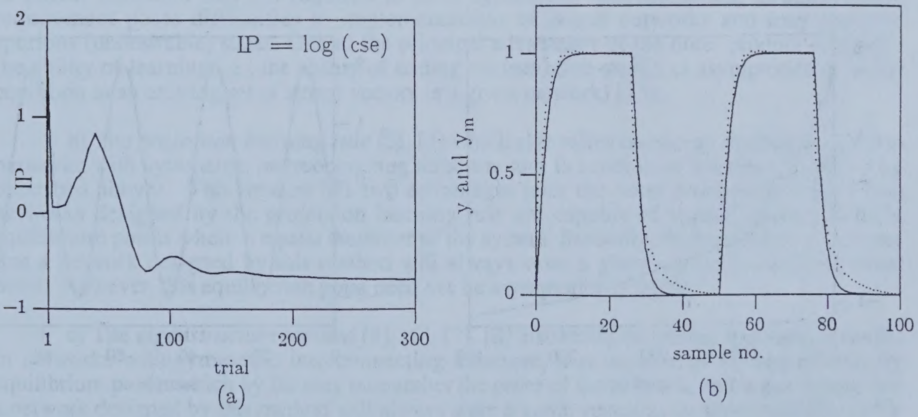

u(k-n)
u(k-1)
y(k-1)
y(k-r)
u_c(k)
u_c(k−s)
Controller
u(k)
Unknown Plant
a_n
a_1
a_0
y(k+1)
g(y(k), ..., y(k-m))
● Nonlinear Neuron
○ Linear Neuron
Input Layer
Hidden Layer
Output Layer

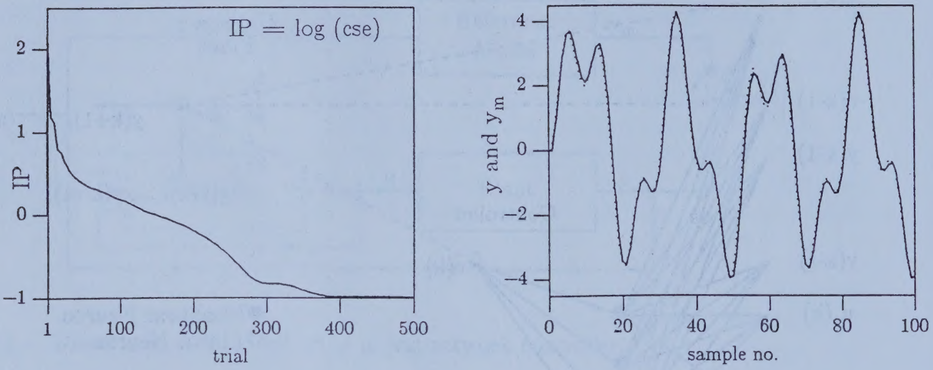
Fig. 3. Controller and plant as a neural network with linear and nonlinear neurons.



(a)

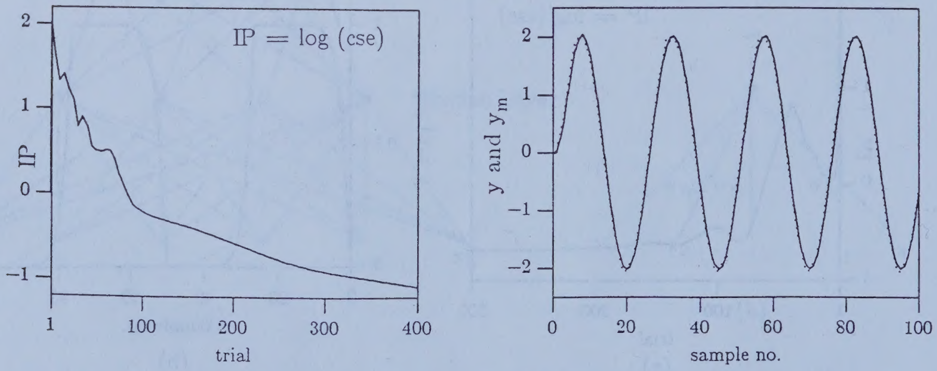
(b)

Fig. 4. Example I : (a) Learning curve (b) Response of the plant and model after 300 trials.



(a) (b)

Fig. 5. Example II : (a) Learning curve (b) Response of the plant and model after 500 trials.



(a) (b)

Fig. 6. Example III : (a) Learning curve (b) Response of the plant and model after 400 trials.

A LEARNING AND FORGETTING ALGORITHM IN ASSOCIATIVE MEMORIES: RESULTS INVOLVING PSEUDO INVERSES*

G. Yen A. N. Michel

Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

The present paper develops a design technique for associative memories with learning and forgetting abilities via artificial feedback neural networks. The proposed method utilizes the theory of large-scale interconnected dynamical systems. Networks synthesized by the design method advanced herein are capable of learning new patterns as well as forgetting old patterns without recomputing the *entire* interconnection matrix. The present results constitute improvements over the outer product method and the projection learning rule. Several specific examples are given to illustrate the strengths and weaknesses of the methodology advocated.

I. INTRODUCTION

Design techniques for associative memories via artificial feedback neural networks have been devised for various models which incorporate the outer product method [1], [2], the projection learning rule [3], [4], the eigenstructure method [5], [6], [7], [8], a method which results in a nonsymmetric interconnecting structure [9], [10], [11], and others. As pointed out in [12], these methods have advantages as well as limitations. Specifically :

a) The *outer product method* [1], [2] does not guarantee that networks will necessarily store desired vectors as equilibrium points of the network. Furthermore, experience has shown that networks designed by the outer product method can store *effectively* only up to 0.15n arbitrary vectors as equilibrium points where n denotes the order of the network. Moreover, networks designed by the outer product method (which makes use of energy functions) result in neural networks that are required to have symmetric interconnecting structure. This requirement poses difficulties in implementations of neural networks and may result in spurious (undesirable) states. One of the principal advantages of the outer product method is the ability of learning(i.e., the ability of adding vectors to be stored as asymptotically stable equilibria to an existing set of stored vectors in a given network) [13].

b) The *projection learning rule* [3], [4] which also relies on energy methods, results in networks with symmetric interconnecting structure and is capable of learning (in the sense described above). This method has two advantages over the outer product method. First, networks designed by the projection learning rule are capable of storing *effectively* 0.5n equilibrium points where n equals the order of the system. Secondly, this technique guarantees that a network designed by this method will always store a given vector as an equilibrium point. However, this equilibrium point need not be asymptotically stable.

c) The *eigenstructure method* [5], [6], [7], [8] also relies on energy methods, it results in networks with symmetric interconnecting structure, it is capable of storing effectively equilibrium points which by far may outnumber the order of the network, and it guarantees that a network designed by this method will always store a given vector as an asymptotically stable equilibrium point. However, the results do not make any provisions for learning.

d) The results given in [9], [10], [11] which are not based on energy methods, are capable of storing up to n (linearly independent) equilibrium points. However, as in the case of

---

*The authors are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana 46556. This work was supported in part by the National Science Foundation under Grant ECS 88-02924.

the projection learning rule, the effectiveness of the attractivity of the stored equilibrium points is greatly reduced as the number of the desired vectors to be stored becomes of order n/2. This method guarantees that a network designed by this technique will always store a given vector as an asymptotically stable equilibrium point. The results in [9], [10], [11] do not make any provisions for learning. However, networks designed by the present method need *not* result in symmetric interconnecting structures. As such, no difficulties are encountered in network implementations.

In addition to *learning* (as defined above), another desirable feature of a neural network design method is *forgetting*(i.e., the ability of deleting specified equilibrium points from a given set of stored equilibria in a given network). Aspects of learning for the case of Hopfield networks designed by the outer product method have been addressed in [13]. The eigenstructure method (discussed above in item (c)) and the results in [9], [10], [11] (discussed above in item (d)) would unquestionably constitute improvements over the outer product method [1], [2] and the projection learning rule [3], [4] (discussed above in items (a) and (b), respectively), if these methods would make provisions for learning and forgetting. In the present paper we address these and other issues by incorporating learning and forgetting capabilities into the design methodology advanced in [9], [10], [11]. In addition to learning and forgetting capabilities, the synthesis method developed in this paper offers significant improvements in computational efficiency.

The structure of the remainder of this paper is as follows. In Section 2, we address the stability analysis of the class of neural networks considered herein. In Section 3, we present a design procedure with learning and forgetting capabilities for this class of networks. The applicability of the present results is demonstrated by means of two specific examples in Section 4. The paper is concluded with a few pertinent observations in Section 5.

II. ANALYSIS

We consider neural network models described by difference equations of the form,

$$u_i(k+1)=\sum_{j=1}^{n} T_{ij}\, v_j(k)+A_{ii}u_i(k)+I_i, \qquad i=1,\ldots,n \qquad (2.1)$$
$$v_i(k)=g(u_i(k)),$$

where $T_{ij}\in R$, $A_{ii}\in R$, $I_i\in R$ and $g: R\rightarrow R$. We assume that g is a monotone increasing, continuously differentiable function with $g(0)=0$. Also, we assume that g satisfies a sector condition, i.e., there are two positive real numbers c_1 and c_2 such that $c_1\leq \frac{g(x)}{x} \leq c_2$ for all $x\in B(r)=\{y\in R : |y|<r\}$ for some $r>0$. The function g serves as the model for all neurons. Thus, u_i and v_i represent the input and output of the ith neuron, respectively. Equation (2.1) can be put into vector form

$$u(k+1)=T\,v(k)+Au(k)+I, \qquad (2.2)$$
$$v(k)=g(u(k)),$$

where $T=[T_{ij}]\in R^{n\times n}$, $A=\mathrm{diag}(A_{ii})\in R^{n\times n}$, $I=[I_i]\in R^n$, and $g=[g,\ldots,g]^T: R^n\rightarrow R^n$.

Any point $u^*\in R^n$ is said to be an equilibrium point of (2.2) if

$$u^*(k)=Tg(u^*(k))+Au^*(k)+I. \qquad (2.3)$$

In analyzing the stability of the equilibrium point u^*, we may assume without loss of generality that $u^*=0$ and $I=0$ to obtain from (2.2), the equivalent system

$$p(k+1)=TG(p(k))+Ap(k), \qquad (2.4)$$

with $G(0)=0$ and with an equilibrium point at the origin, $p(k)=0$. Equation (2.4) can equivalently be expressed as

$$p_i(k+1)=\sum_{j=1}^{n} T_{ij}\,G(p_j(k))+A_{ii}p_i(k), \qquad i=1,\ldots,n. \tag{2.5}$$

System (2.5) may be viewed as an interconnection of n free subsystems represented by equation (2.6)

$$x_i(k+1)=T_{ii}G(x_i(k))+A_{ii}x_i(k), \tag{2.6}$$

$i=1,\ldots,n$. The interconnecting structure of system (2.5) is specified by

$$h_i(x_1(k),\ldots,x_n(k)) \triangleq \sum_{\substack{j=1\\ j\neq i}}^{n} T_{ij}\,G(x_j(k)), \tag{2.7}$$

$i=1,\ldots,n$. To analyze the stability properties of the equilibrium point $p(k)=0$ for system (2.4), we need the following hypotheses.

Assumption 2.1. For each free subsystem (2.6),

$$0<\sigma_i \triangleq (|A_{ii}|+|T_{ii}|c_2)<1$$

where c_2 defines the sector condition for the function $g(\cdot)$. ☐

It is shown in [9], [11] that in neural network applications, Assumption 2.1 is reasonable and realistic. Also, it is shown in [9], [11] that if Assumption 2.1 is true, then the equilibrium point $x_i=0$ of (2.6) is asymptotically stable.

Assumption 2.2. Give σ_i of Assumption 2.1, the successive principal minors of the matrix $D=[D_{ij}]\in R^{n\times n}$ are all positive, where

$$D_{ij}=\begin{cases} -(\sigma_i-1) & \text{when } i=j \\ -\sigma_{ij} & \text{when } i\neq j \end{cases},$$

where $\sigma_{ij}=|T_{ij}|c_2$ (c_2 is given in Assumption 2.1). ☐

It is shown in [9], [11] that if Assumptions 2.1 and 2.2 are true, then the equilibrium point $p(k)=0$ of neural network (2.4) is asymptotically stable. Note that if Assumption 2.2 is true, then matrix D is an M-matrix. For some of the important properties of M-matrices, refer to [14]. In particular, it can be shown that Assumption 2.3 implies Assumption 2.2.

Assumption 2.3. For the matrix $D=[D_{ij}]$ (as defined in Assumption 2.2), there exist constants $\lambda_i>0$, $i=1,\ldots,n$, such that

$$D_{ii}+\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\lambda_j}{\lambda_i}D_{ji}\geq \varepsilon>0, \qquad i=1,\ldots,n. \quad ☐$$

It is shown in [9], [11] that if Assumption 2.3 is true, then

$$\|u(k)-u^*\| = \|p(k)\| \leq \mu^k\,\|p(0)\| \tag{2.8}$$

for all $\|p\|<\min_i(\lambda_i,r)$, where $\|p\|=\sum_{i=1}^{n}\lambda_i\,|p_i|$, r is specified in the definition of the sector condition of the function $g(\cdot)$, the constants λ_i, $i=1,\ldots,n$ are given in Assumption 2.3, and $0<\mu\triangleq 1-\varepsilon<1$.

The estimate of the trajectory bounds given in (2.8) provides us with an indication of the rate of convergence of a trajectory near an equilibrium point u* of (2.4). For nonsymmetric interconnection matrices T, the present results do not guarantee that neural network (2.4) is globally stable(i.e., that *all* trajectories of system (2.4) tend to *some* equilibrium point). However, of the numerous simulation runs performed in [10] and [11] for nonsymmetric T, nearly all of the trajectories (typically 99%) converged to some equilibrium. (It can be argued that global stability is not necessarily a desirable feature for a neural network, since for such systems, trajectories may tend to incorrect equilibria (spurious states). On the other hand, if in a network which is not globally stable, a given trajectory does not converge, this knowledge may be put to good use.)

III. SYNTHESIS

The present section is divided into four parts. In Subsection A, we provide a summary of a synthesis procedure which was developed in [11]. This procedure is based on the analysis method established in Section 2. In Subsection B, we develop a new design technique for associative memories which incorporates learning capabilities into the method advanced in [11]. In Subsection C, we modify the results of Subsection B in order to develop an on-line iterative learning algorithm for the class of neural networks considered herein. Finally, in Subsection D we devise a technique which enables us to incorporate forgetting capabilities into the method advocated in [11].

A. A Synthesis Procedure for Neural Networks

The design procedure developed in [11] involves two stages. In the *first step* one chooses the interconnections $T=[T_{ij}]$, the internal self-feedback terms $A=[A_{ii}]$, and the external inputs $I=[I_i]$ in order to store the patterns $v^i=g(\lambda u^i)$, $i=1,\ldots,m$ corresponding to equilibrium points u^i, $i=1,\ldots,m$ of the neural network (λ denotes a network parameter, (the "gain" of the neurons).). In the *second step*, the network is manipulated so that each of the patterns v^i, $i=1,\ldots,m$ corresponds to an asymptotically stable equilibrium point u^i, $i=1,\ldots,m$. Let $\Delta=\{v^1,\ldots,v^m\}$ denote the set of desired library vectors to be stored as asymptotically stable equilibrium points of the neural network. Each $v^i\in\Delta$ corresponds to an equilibrium point of (2.2), if and only if

$$u^i=Tv^i+Au^i+I, \tag{3.1}$$

$i=1,\ldots,m$, where $u^i=[u^i_1,\ldots,u^i_n]^T\in R^n$ and $v^i=[v^i_1,\ldots,v^i_n]^T\in R^n$.

To present the design algorithm in compact form, we define the matrices

$$L_m=[v^1,\ldots,v^m], \tag{3.2a}$$

$$H_m=[u^1,\ldots,u^m], \tag{3.2b}$$

where the subscript m for L_m denotes the number of vectors to be stored in the network. Equation (3.1) can now be expressed as

$$H_m=TL_m+AH_m+\tilde{I} \tag{3.3}$$

where $\tilde{I}=[I,\ldots,I]\in R^{n\times m}$. Our goal is to determine the matrices T, A and the vector I, so that constraint (3.3) is satisfied. We let

$$R_{m,j}=[L_m^T\ H_{m,j}^T\ Q_m], \qquad j=1,\ldots,n, \tag{3.4a}$$

$$W_{m,j}=[T_{j1},\ldots,T_{jn}, A_{jj}, I_j], \qquad j=1,\ldots,n \tag{3.4b}$$

where $H_{m,j} \in R^m$ is the jth row of H_m, and $Q_m \in R^m$ is a column vector containing all ones. Solving (3.3) is equivalent to solving

$$H^T_{m,j} = R_{m,j}\, W^T_{m,j}\,, \qquad j=1,\dots,n \tag{3.5}$$

where matrix $R_{m,j} \in R^{m\times(n+2)}$ and row vector $W_{m,j} \in R^{n+2}$. Equation (3.5) has a well-known solution involving the pseudoinverse matrix [15], [16], given by

$$\begin{aligned} W^T_{m,j} &= R^+_{m,j}\, H^T_{m,j}, \qquad j=1,\dots,n \qquad (3.6)\\ &= R^T_{m,j}\, (R_{m,j}\, R^T_{m,j})^{-1}\, H^T_{m,j} \end{aligned}$$

where $R^+_{m,j}$ is the pseudoinverse of $R_{m,j}$. We usually assume $m \le n$ to guarantee the effectiveness of the attractivity of the stored equilibrium points. $R_{m,j}$ therefore possesses the right inverse with its columns in the column space of $R^T_{m,j}$. The solution (3.6) is unique. Constraint (3.5) allows us to design a network with each element of Δ as an equilibrium point. The following constraint will guarantee that each element of Δ will be asymptotically stable :

$$F_j = 1 - |A_{jj}| - \sum_{i=1}^{n} |T_{ji}| \ge 0 \qquad j=1,\dots,n. \tag{3.7}$$

Constraint (3.7) is an immediate consequence of Assumption 2.3, by letting $\lambda_i=1$, i=1,...,n.

B. Learning Algorithm

Now suppose that a network (2.2) has been designed in such a manner that the vectors $\{v^1,\dots,v^m\}$ are stored as asymptotically stable equilibrium points and suppose we desire to store an additional pattern $v^{m+1}=g(\lambda u^{m+1})$ corresponding to an equilibrium point u^{m+1}. Define

$$L_{m+1}=[v^1,\dots,v^m,\ v^{m+1}]=[L_m\ \ v^{m+1}], \tag{3.8a}$$
$$H_{m+1}=[u^1,\dots,u^m,\ u^{m+1}]=[H_m\ \ u^{m+1}]. \tag{3.8b}$$

Let $H_{m+1,j} \in R^{m+1}$ denote the jth row of H_{m+1}. Then

$$H_{m+1,j} = \begin{bmatrix} H_{m,j} & u_j^{m+1} \end{bmatrix}, \qquad j=1,\dots,n, \tag{3.9a}$$

$$\begin{aligned} R_{m+1,j} &= [L^T_{m+1}\ \ H^T_{m+1,j}\ \ Q_{m+1}], \qquad j=1,\dots,n \qquad (3.9b)\\ &= \begin{bmatrix} L^T_m & H^T_{m,j} & Q_m \\ (v^{m+1})^T & u_j^{m+1} & 1 \end{bmatrix} = \begin{bmatrix} R_{m,j} \\ M_{m,j} \end{bmatrix} \end{aligned}$$

where $M_{m,j} = \begin{bmatrix} (v^{m+1})^T & u_j^{m+1} & 1 \end{bmatrix} \in R^{n+2}$ and $Q_{m+1}=[Q^T_m\ \ 1]^T$. In order to satisfy the constraint (3.1) for i=1,...,m+1, we need to solve equation (3.5) and an additional equation

$$u^{m+1} = Tv^{m+1} + Au^{m+1} + I. \tag{3.10}$$

This is equivalent to solving the vector equation given by

$$H^T_{m+1,j} = R_{m+1,j}\, W^T_{m+1,j}, \qquad j=1,\dots,n \tag{3.11}$$

where $W_{m+1,j} \in R^{n+2}$ is the resulting vector we desire to find after we have successfully stored an additional pattern(i.e., v^{m+1}). Utilizing the results for the pseudoinverse of a partitioned matrix, we obtain the following statements. (The detailed proofs of these results will be given elsewhere.)

$$W^T_{m+1,j} = R^+_{m+1,j}\, H^T_{m+1,j}, \qquad j=1,\dots,n \tag{3.12}$$

$$= W^T_{m,j} - b^T_m M_{m,j} W^T_{m,j} + b^T_m u^{m+1}_j$$

where $b^T_m = (M_{m,j} - M_{m,j} R^+_{m,j} R_{m,j})^+$, if $(I - R^T_{m,j}(R^T_{m,j})^+)M^T_{m,j} \neq 0$,

$b^T_m = (R^T_{m,j} R_{m,j})^{-1} M^T_{m,j} (1 + M_{m,j} (R^T_{m,j} R_{m,j})^{-1} M^T_{m,j})^{-1}$,

if $(I - R^T_{m,j}(R^T_{m,j})^+)M^T_{m,j} = 0$.

The subsequent derivations will be simplified considerably by assuming that the neural network (2.1) is not endowed with self-feedback terms. With this assumption, by interchanging the last two components of $W_{m+1,j}$ and the last two columns of $R_{m+1,j}$, we obtain $M_{m,j}=M_m$, $\forall j$. Putting equation (3.12) into vector form, we have

$$W^T_{m+1} = W^T_m + \Delta W^T_m \tag{3.13}$$

$$= W^T_m - b^T_m M_m W^T_m + b^T_m (u^{m+1})^T,$$

where $\Delta W^T_m = b^T_m[(u^{m+1})^T - M_m W^T_m]$.

The preceding results enable us to learn an additional pattern v^{m+1} without affecting the patterns $v^1,\dots v^m$ already learned by the network. (However, this process may result in a decrease of the extent of the domain of attraction for the existing asymptotically stable equilibrium points $u^1,\dots,u^m$.) At this point, we have not yet ascertained the asymptotic stability of the equilibrium u^{m+1}. This will be accomplished next.

Initially, the designer chooses a gain λ_0 for the input-output relationship $v^{m+1} = g(\lambda_0 u^{m+1})$. Using this gain the designer determines matrix $M_{m,j}$ and vector b^T. The initial solution of (3.12) for this choice of λ_0 will be denoted by $W^0_{m+1,j}$, $j=1,\dots,n$. Define

$$\varepsilon = \min_{1 \le j \le n} (F_j(W^0_{m+1,j})) \tag{3.14}$$

where $F_j = 1 - |A_{jj}| - \sum_{i=1}^{n} |T_{ji}|$.

If $\varepsilon \ge 0$, the constraints (3.7) and (3.11) are both satisfied. The desired values of T, A and I are determined by the $W^0_{m+1,j}$, $j=1,\dots,n$. If $\varepsilon<0$, choose a constant $\delta > 1-\varepsilon > 1$, and define

$$\lambda = \delta\lambda_0. \tag{3.15}$$

Then, $W_{m+1,j} = W^0_{m+1,j}/\delta$ will satisfy the constraints (3.7) and (3.11).

C. Iterative Learning Algorithm

The above algorithm can be modified to enable us to store (learn) iteratively *all* of the desired equilibrium points $u^1,\dots,u^m$. This procedure supersedes the design procedure given in [11] since in addition to learning capabilities, it also improves computational efficiency. The *iterative learning algorithm* can be summarized as follows :

Step 1 : to learn the pattern v^1(i.e., $\Delta=\{v^1\}$), let

$$W_1^T=R_1^+ H_1^T \tag{3.16a}$$

$$=[(v^1)^T \quad 1]^+ (u^1)^T.$$

Step 2 : to learn the pattern v^2(i.e., $\Delta=\{v^1,v^2\}$), let

$$W_2^T=W_1^T + \Delta W_1^T, \tag{3.16b}$$

where $\Delta W_1^T=b_1^T[(u^2)^T - M_1 W_1^T]$, $b_1^T=(M_1 - M_1 R_1^+ R_1)^+$, and $M_1=[(v^2)^T \quad 1]$.

……

Step i : to learn the pattern v^i(i.e., $\Delta=\{v^1,v^2,\dots,v^i\}$), let

$$W_i^T=W_{i-1}^T + \Delta W_{i-1}^T, \tag{3.16c}$$

where $\Delta W_{i-1}^T=b_{i-1}^T[(u^i)^T - M_{i-1} W_{i-1}^T]$, $b_{i-1}^T=(M_{i-1} - M_{i-1} R_{i-1}^+ R_{i-1})^+$, $M_{i-1}=[(v^i)^T \quad 1]$,

$$R_{i-1}=\begin{bmatrix} R_{i-2} \\ M_{i-2} \end{bmatrix} \quad \text{and} \quad R_{i-1}^+=[R_{i-2}^+ - b_{i-2}^T M_{i-2} R_{i-2}^+ \quad b_{i-2}^T].$$

The above algorithm is applicable when there are *no prerepresentations* for the neural network. *Note that the design of a neural network by the above algorithm involves no matrix inversions.*

D. Forgetting Algorithm

Now suppose that we desire to forget a known pattern $v^i \in \Delta$ corresponding to an equilibrium point u^i. We have

$$R_m =[L_m^T \; Q_m \; Z_m] \tag{3.17}$$

$$=\begin{bmatrix} L_{m-1}^T & Q_{m-1} & Z_{m-1} \\ (v^i)^T & 1 & 0 \end{bmatrix} = \begin{bmatrix} R_{m-1} \\ M_{m-1} \end{bmatrix}.$$

Rearranging the matrices L_m and H_m by interchanging vectors v^i, u^i with the vectors v^m, u^m, respectively (if necessary), we obtain the matrices

$$\tilde{L}_m =[v^1,\dots,v^{i-1},v^{i+1},\dots,v^m,v^i]=[L_{m-1} \quad v^i], \tag{3.18a}$$

$$\tilde{H}_m =[u^1,\dots,u^{i-1},u^{i+1},\dots,u^m,u^i]=[H_{m-1} \quad u^i]. \tag{3.18b}$$

Let b_{m-1} denote the i^{th} column of matrix R_m^+, and let S_{m-1} denote the matrix R_m^+ with b_{m-1} removed. From (3.12), we now obtain after some manipulations

$$S_{m-1}=(R_{m-1}^+)^T(E_{n+2} - M_{m-1}^T b_{m-1}),$$

$$R^{+}_{m-1}=(E_{n+2}-b^{T}_{m-1}M_{m-1})^{-1}S^{T}_{m-1}. \tag{3.19}$$

The desired network parameters are now determined by

$$W^{T}_{m-1}=R^{+}_{m-1}H^{T}_{m-1}. \tag{3.20}$$

IV. EXAMPLES

To demonstrate the applicability and efficiency of the synthesis results established in Section 3, we now consider two specific examples. In the first example we assume a given prerepresentation for the network. In the second example the network is assumed to be void of any prerepresentation.

Example 4.1. The dimension of the system to be considered is n=10 and we assume that the m=4 vectors are specified by

$$v^1=\begin{bmatrix}-1\\1\\1\\1\\1\\1\\-1\\-1\\-1\\-1\end{bmatrix}, \qquad v^2=\begin{bmatrix}1\\1\\-1\\-1\\-1\\1\\-1\\-1\\1\\-1\end{bmatrix}, \qquad v^3=\begin{bmatrix}-1\\1\\1\\1\\-1\\-1\\1\\-1\\-1\\-1\end{bmatrix}, \qquad v^4=\begin{bmatrix}-1\\1\\-1\\-1\\-1\\-1\\1\\-1\\1\\1\end{bmatrix}.$$

By following the learning procedure described in Section 3, the network desires to learn v^5 and v^6 given by

$$v^5=\begin{bmatrix}1\\-1\\-1\\1\\1\\-1\\1\\1\\1\\-1\end{bmatrix}, \qquad v^6=\begin{bmatrix}1\\1\\-1\\1\\-1\\1\\1\\1\\-1\\-1\end{bmatrix}.$$

It is required that after the learning phase, all of the vectors $v^1,\dots,v^6$ be stored as asymptotically stable equilibrium points. The algorithm was successfully applied to the above problem, as verified by digital simulations of the resulting synthesized network. In addition to $\{v^1,\dots,v^6\}$, the resulting network also stored the following nine vectors as asymptotically stable equilibrium points(i.e., spurious states),

$$\alpha^1=\begin{bmatrix}1\\1\\-1\\-1\\-1\\-1\\1\\-1\\1\\-1\end{bmatrix},\alpha^2=\begin{bmatrix}-1\\1\\-1\\-1\\1\\1\\-1\\-1\\1\\1\end{bmatrix},\alpha^3=\begin{bmatrix}-1\\1\\-1\\1\\1\\1\\1\\1\\-1\\1\end{bmatrix},\alpha^4=\begin{bmatrix}-1\\-1\\1\\1\\1\\-1\\-1\\-1\\1\\-1\end{bmatrix},\alpha^5=\begin{bmatrix}-1\\-1\\-1\\1\\1\\-1\\1\\1\\1\\1\end{bmatrix},\alpha^6=\begin{bmatrix}1\\-1\\-1\\-1\\1\\-1\\-1\\-1\\1\\-1\end{bmatrix},\alpha^7=\begin{bmatrix}1\\-1\\-1\\1\\1\\1\\-1\\1\\1\\-1\end{bmatrix},\alpha^8=\begin{bmatrix}-1\\1\\1\\-1\\-1\\-1\\-1\\-1\\1\\-1\end{bmatrix},\alpha^9=\begin{bmatrix}1\\1\\-1\\-1\\-1\\1\\1\\1\\1\\1\end{bmatrix}.$$

Continuing, we next apply the forgetting algorithm to the above synthesized network. Our objective is to forget(i.e., delete) v^3 and v^5 from the library vectors of the synthesized system. Once more, the algorithm was successfully applied, as verified by simulations: v^3 and v^5 were deleted from the network as asymptotically stable equilibrium points while v^1, v^2, v^4

and v^6 were retained as asymptotically stable equilibrium points. The resulting network stored only the following vectors as spurious states,

$$\beta^1 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \quad \beta^2 = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}.$$

The choice of n=10 in the above example was influenced by two factors : on the one hand, we wanted to avoid a trivial problem (by selecting n very small); on the other hand, we wished to avoid the distraction of numerical complexities of a very high dimensional network. In the above example, the desired vectors to be stored(i.e., $v^1,...,v^4$) as well as the vectors to be learned(i.e., v^5 and v^6) were randomly chosen with each bit having probability equal to 1/2 of being 1 or -1 by calling the IMSL routine RNUN. □

Example 4.2. Utilizing the *iterative learning algorithm* developed in Section 3.C, we designed several networks(i.e., storing different sets of vectors $\{v^1,...,v^6\}$) with n=10. In all cases, the algorithm performed as expected.

The algorithm has been implemented in a manner which makes possible the design of a network in an *iterative mode*. As mentioned earlier, in addition to *on-line* iterative design capability, the present algorithm offers a significant reduction in numerical complexity (compared to the results in [11]), as is illustrated in Figures 4.1 and 4.2. The original method without learning utilizes the Gauss-Jordan elimination technique with $O(n^3)$ computational complexity in determining required matrix inversions.

It is important that Figures 4.1 and 4.2 be interpreted correctly. This requires that the values on the vertical axes of the plots without learning be read absolutely while the values on the vertical axes of the plots with learning be interpreted incrementally. For example, if six vectors have been stored in the present network (with n=10) and if it is desired to learn an additional vector, then in accordance with Figures 4.1 and 4.2, the algorithm without learning will require 3350 real multiplications and 3097 real additions while the algorithm with learning will require only 1927-1553=374 real multiplications and 1831-1475=356 real additions. □

We conclude the present section by noting that all of the above examples were simulated on a digital computer (Sun Sparcstation 1) with less than 100 lines of FORTRAN code.

V. CONCLUSIONS

We offer a few pertinent observations concerning the strengths and weaknesses of the design method presented herein.

The synthesis procedure for neural networks advanced in [9]-[11] has several attractive features : it guarantees that desired sets of vectors be stored as asymptotically stable equilibrium points; it yields networks with interconnecting structure which need not be symmetric; and the storing capacity of networks designed by this procedure exceeds the storing capacity of some of the more popular synthesis techniques(e.g., the outer product method). However, the design procedure developed in [9]-[11] has also two principal shortcomings: networks generated by this method are not necessarily globally stable; and this method has no provisions for learning and forgetting.

In the present paper we built on the results given in [9]-[11] to develop a synthesis procedure for neural networks with learning and forgetting capabilities. Learning and forgetting algorithms and an iterative learning algorithm were developed. In addition to providing on-line learning capabilities and other flexibilities, the present results offer significant improvements in computational efficiency.

REFERENCES

[1] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. Natl. Acad. Sci., U.S.A., Vol. 79, April 1982, pp. 2554-2558.

[2] J. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons", Proc. Natl. Acad. of Sci., U.S.A., Vol. 81, May 1984, pp. 3088-3092.

[3] L. Personnaz, I. Guyon and G. Dreyfus, "Information Storage and Retrieval in Spin-Glass Like Neural Networks", J. Physique Letters, Vol. 46, April 1985, pp. L359-365.

[4] L. Personnaz, I. Guyon and G. Dreyfus, "Collective Computational Properties of Neural Networks: New Learning Mechanisms", Physical Review A, Vol. 34, No. 5, November 1986, pp. 4217-4228.

[5] J. H. Li, A. N. Michel and W. Porod, "Qualitative Analysis and Synthesis of A Class of Neural Networks", IEEE Trans. on Circuits and Systems, Vol. 35, No. 8, August 1988, pp. 976-986.

[6] J. H. Li, A. N. Michel and W. Porod, "Analysis and Synthesis of A Class of Neural Networks: Variable Structure Systems with Infinite Gains", IEEE Trans. on Circuits and Systems, Vol. 36, No. 5, May 1989, pp. 713-731.

[7] J. H. Li, A. N. Michel and W. Porod, "Analysis and Synthesis of A Class of Neural Networks: Linear Systems Operating on A Closed Hypercube", IEEE Trans. on Circuits and Systems, Vol. 36, No. 11, November 1989, pp. 1405-1422.

[8] A. N. Michel, J. Si and G. Yen, "Analysis and Synthesis of A Class of Discrete-Time Neural Networks Described on A Closed Hypercube", Proc. 1990 IEEE Symposium on Circuits and Systems, New Orleans, Louisiana, May 1990, pp. 700-703.

[9] A. N. Michel, J. A. Farrell and W. Porod, "Qualitative Theory of Neural Networks", IEEE Trans. on Circuits and Systems, Vol. 36, No. 2, February 1989, pp. 229-243.

[10] J. A. Farrell and A. N. Michel, "A Synthesis Procedure for Hopfield's Continuous-Time Associative Memory", IEEE Trans. on Circuits and Systems, Vol. 37, No. 7, July 1990, pp. 877-884.

[11] A. N. Michel, J. A. Farrell and H. F. Sun, "Analysis and Synthesis Techniques for Discrete-Time Neural Network Models", Proc. 1989 IEEE Conference on Decision and Control, Tampa, Florida, December 1989, pp. 773-778.

[12] A. N. Michel and J. A. Farrell, "Associative Memories via Artificial Neural Networks", IEEE Control Systems Magazine, Vol. 10, No. 3, April 1990, pp. 6-17.

[13] J. J. Hopfield, D. I. Feinstein and R. E. Palmer, " 'Unlearning' Has A Stability Effect in Collective Memories", Nature, Vol. 304, No. 5922, July 1983, pp. 158-159.

[14] A. N. Michel and R. K. Miller, Qualitative Analysis of Large Scale Dynamical Systems, Academic Press, New York, 1977.

[15] A. Albert, Regression and the Moore-Penrose Pseudo-Inverse, Academy Press, New York, 1972.

[16] T. N. E. Greville, "The Pseudoinverse of A Rectangular or Singular Matrix and Its Application to the Solution of Systems of Linear Equations", SIAM Review, Vol. 1, No. 1, January 1959, pp. 38-43.

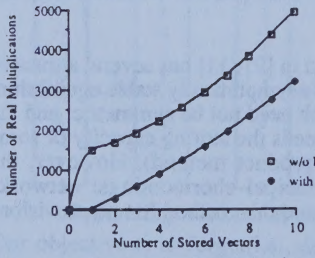


Figure 4.1. Number of real multiplications required for the design methods with/without learning ability

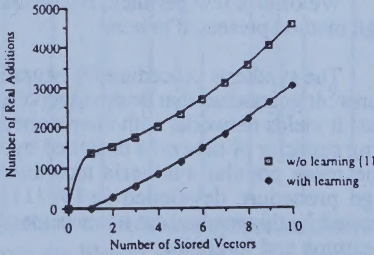


Figure 4.2. Number of real additions required for the design methods with/without learning ability

UNIVERSAL APPROXIMATION USING RADIAL-BASIS-FUNCTION NETWORKS*

J. Park and I. W. Sandberg

Department of Electrical and Computer Engineering,
University of Texas at Austin
Austin, Texas 78712

Abstract

There have been several recent studies concerning feedforward networks and the problem of approximating arbitrary functionals of a finite number of real variables. Some of these studies deal with cases in which the hidden-layer nonlinearity is not a sigmoid. This was motivated by successful applications of feedforward networks with non-sigmoidal hidden-layer units.

This paper reports on a related study of radial-basis-function (RBF) networks, and it is proved that RBF networks having one hidden-layer are capable of universal approximation. Here the emphasis is on the case of typical RBF networks, and the results show that a certain class of RBF networks with the same smoothing factor in each kernel node is broad enough for universal approximation.

1 Introduction

There have been several recent studies concerning the capabilities of multilayered feedforward neural networks. Particularly pertinent to this paper are results which show that certain classes of neural networks are capable of providing arbitrarily good approximations to prescribed functionals of a finite number of real variables. From the theoretical point of view, these studies are important, because they address the question of whether a satisfactory solution is yielded by some member of a given class of networks. More specifically, suppose we have a problem that we want to solve using a certain type of neural network. Suppose also that there exists a decision function $f : \Re^r \rightarrow \Re^m$ whose implementation as a network plays a central role in the solution of the problem. Imagine that we have a family G of functions mapping $\Re^r$ to $\Re^m$ characterized by a certain structure and having certain elements (e. g. one might consider a set of multilayered perceptrons), and that we hope to solve the problem by implementing some satisfactory member of G. The first question we need to consider

---

*This work was supported in part by the National Science Foundation under Grant MIP-8915335.

might be: Is this family G broad enough to contain f or a good approximation of f? Obviously, attempts to solve the problem without considering this question might be very time-consuming and might even be fruitless.

Several papers address this question for the case of multilayered perceptron models with sigmoidal nonlinearities, and affirmative answers have been obtained by showing that in a satisfactory sense the family G considered can actually approximate any decision function drawn from a certain large class [1,2].

At the present time, with the advantages and limitations of multilayered perceptron networks more transparent and with results containing comparative studies becoming available(e. g. [3]), research concerning different types of feedforward networks is very active. Among the various kinds of promising networks are the so-called radial-basis-function (RBF) networks [3]. The block diagram of a version of an RBF classifier with one hidden-layer is shown in Fig. 1. Each unit in the hidden-layer of this RBF network has its own centroid, and for each input $x = (x_1, x_2, \cdots, x_r)$, it computes the distance between x and its centroid. Its output (the output signal at one of the kernel nodes) is some nonlinear function of that distance. Thus, each kernel node in the RBF network computes an output that depends on a radially symmetric function, and usually the strongest output is obtained when the input is near the centroid of the node.

Assuming that there are r input nodes and m output nodes, the overall response function without considering nonlinearity in an output node has the following form:

$$\sum_{i=1}^{M} W_i \cdot K(\frac{x - z_i}{\sigma_i}) = \sum_{i=1}^{M} W_i \cdot g(\frac{\| x - z_i \|}{\sigma_i}), \tag{1}$$

where $M \in \mathcal{N}$ the set of natural numbers is the number of kernel nodes in the hidden-layer, $W_i \in \Re^m$ is the vector of weights from the i^{th} kernel node to the output nodes, x is an input vector (an element of $\Re^r$), K is a radially symmetric kernel function of a unit in the hidden-layer, z_i and σ_i are the centroid and smoothing factor (or width) of the i^{th} kernel node, respectively, and $g : [0, \infty) \rightarrow \Re$ is a function called the activation function, which characterizes the kernel shape.

A Gaussian function is often used as an activation function, and the smoothing factors of kernel nodes may be the same or may vary across nodes.

In this paper, RBF networks having the representation (1) are studied. Strong results are obtained to the effect that, under certain mild conditions on the kernel function K (or the activation function g), RBF networks represented by (1) with the same σ_i in each kernel node have the capability of universal approximation. Ref. 1 also considers feedforward networks with a single hidden-layer of kernel functions. However, only L^1 approximation is considered in the corresponding part of [1], and only the case in which the smoothing factors can vary across nodes is addressed. A detailed comparison is given in Section 3.

This paper is organized as follows: In Section 2 our main results are presented, and in Section 3 a discussion of our results is given.

2 Main Results

In this section, we consider the approximation of a function by some element of a specific family of RBF networks.

Throughout the paper, we use the following notation and definitions, in which $\mathcal{N}$, $\Re$ and $\Re^r$ denote the set of natural numbers, the set of real numbers and the set of real r-vectors, respectively. Let $L^p(\Re^r)$, $L^\infty(\Re^r)$, $C(\Re^r)$, and $C_c(\Re^r)$, respectively, denote the usual spaces of $\Re$-valued maps f defined on $\Re^r$ such that f is p^{th} power integrable, essentially bounded, continuous, and continuous with compact support. The usual L^p and L^∞ norms are denoted by $\| \cdot \|_p$ and $\| \cdot \|_\infty$, respectively. The integral of $f \in L^1(\Re^r)$ over a Lebesgue measurable set A in $\Re^r$ is written as $\int_A f(x)dx$ or, if f is a function of several variables and, say, $f(a, \cdot) \in L^1(\Re^r)$ we write $\int_A f(a,x)dx$ to denote the integral of $f(a, \cdot)$ over A. The convolution operation is denoted by " $*$ ", and the characteristic function of a Lebesgue measurable subset A of $\Re^r$ is written as 1_A.

The family of RBF networks considered here consists of functions $q : \Re^r \to \Re$ represented by

$$q(x) = \sum_{i=1}^{M} w_i \cdot K\left(\frac{x - z_i}{\sigma}\right), \tag{2}$$

where $M \in \mathcal{N}$, $\sigma > 0$, $w_i \in \Re$, and $z_i \in \Re^r$ for $i = 1, \cdots, M$. We call this family S_K.

Note that (2) is same as (1), with the exception that the smoothing factors in all kernel nodes are same, and the output space is $\Re$ instead of $\Re^m$. It will become clear that the extension of our results to multidimensional output spaces is trivial, and so we consider only a one dimensional output space.

We will use the following result, which is a slight modification of a theorem in [5, p.101].

Lemma 1: Let $f \in L^p(\Re^r)$, $p \in [1, \infty)$, and let $\phi : \Re^r \to \Re$ be an integrable function such that $\int_{\Re^r} \phi(x)dx = 1$. Define $\phi_\epsilon : \Re^r \to \Re$ by $\phi_\epsilon(x) = \frac{1}{\epsilon^r}\phi(\frac{x}{\epsilon})$ for $\epsilon > 0$. Then $\| \phi_\epsilon * f - f \|_p \to 0$ as $\epsilon \to 0$.

Proof:

Note that $\phi_\epsilon \in L^1(\Re^r)$. By a direct extension from $\Re$ to $\Re^r$ of a standard theorem in analysis [5, p.99], one has $\phi_\epsilon * f \in L^p(\Re^r)$ which is used below.

By a change of variable,

$$(\phi_\epsilon * f)(\alpha) = \int_{\Re^r} f(\alpha - x)\phi_\epsilon(x)dx = \int_{\Re^r} f(\alpha - \epsilon x)\phi(x)dx.$$

Thus,

$$| (\phi_\epsilon * f)(\alpha) - f(\alpha) | \;=\; | \int_{\Re^r} [f(\alpha - \epsilon x) - f(\alpha)]\phi(x)dx \,| \,.$$

With q defined by $\frac{1}{p} + \frac{1}{q} = 1$,

$$\| \phi_\epsilon * f - f \|_p \leq \sup_{h \in L^q(\Re^r), \|h\|_q = 1} \int_{\Re^r} | h(\alpha) | \int_{\Re^r} | f(\alpha - \epsilon x) - f(\alpha) | \cdot | \phi(x) | \,dx d\alpha$$

$$= \sup_{h \in L^q(\Re^r), \|h\|_q = 1} \int_{\Re^r} | \phi(x) | \int_{\Re^r} | h(\alpha) | \cdot | f(\alpha - \epsilon x) - f(\alpha) | \,d\alpha dx$$

$$\leq \int_{\Re^r} \mid \phi(x) \mid \cdot \parallel f(\cdot - \epsilon x) - f(\cdot) \parallel_p dx,$$

by Fubini's theorem and Holder's inequality.

Since $\parallel f(\cdot - \epsilon x) - f(\cdot) \parallel_p \leq 2 \parallel f \parallel_p$ and translation is continuous in $L^p(\Re^r)$ (see [5, p.98] and consider its direct extension to $\Re^r$) we have

$$\parallel \phi_\epsilon * f - f \parallel_p \rightarrow 0 \text{ as } \epsilon \rightarrow 0,$$

by Lebesgue's dominated convergence theorem. This proves the lemma.

Our Theorem 1 (below) establishes that, under certain mild conditions on the kernel function K, RBF networks represented by (2) are capable of approximating arbitrarily well any function in $L^p(\Re^r)$.

Theorem 1: Let $K : \Re^r \rightarrow \Re$ be an integrable bounded function such that K is continuous almost everywhere and $\int_{\Re^r} K(x)dx \neq 0$. Then the family S_K is dense in $L^p(\Re^r)$ for every $p \in [1, \infty)$.

Proof:

Let $p \in [1, \infty), f \in L^p(\Re^r)$, and $\epsilon > 0$.

Since $C_c(\Re^r)$ is dense in $L^p(\Re^r)$ [6, p.69], there exists an $f_c \in C_c(\Re^r)$ such that $\parallel f_c - f \parallel_p < \epsilon/3$. We will assume below that f_c is nonzero. Notice that this involves no loss of generality.

Let $\phi : \Re^r \rightarrow \Re$ be defined by $\phi(x) = \frac{1}{\int_{\Re^r} K(\alpha)d\alpha} \cdot K(x)$, for $x \in \Re^r$. Then ϕ satisfies the conditions on ϕ in Lemma 1. Thus, by defining $\phi_\sigma : \Re^r \rightarrow \Re$ as in Lemma 1, we obtain $\parallel \phi_\sigma * f_c - f_c \parallel_p \rightarrow 0$ as $\sigma \rightarrow 0$. Therefore, there is a positive $\sigma > 0$ such that $\parallel \phi_\sigma * f_c - f_c \parallel_p < \epsilon/3$.

Since f_c has compact support, there exists a positive T such that $\text{supp} f_c \subset [-T, T]^r$. Note that $\phi_\sigma(\alpha - \cdot) f_c(\cdot)$ is Riemann integrable on $[-T, T]^r$, because it is continuous almost everywhere and is bounded by $\parallel \phi_\sigma \parallel_\infty \cdot \parallel f_c \parallel_\infty$.

Define $v_n : \Re^r \rightarrow \Re$ by

$$v_n(\alpha) = \sum_{i=1}^{n^r} \phi_\sigma(\alpha - \alpha_i) f_c(\alpha_i) (\frac{2T}{n})^r,$$

where the set $\{\alpha_i \in \Re^r : i = 1, 2, \cdots, n^r\}$ consists of all points in $[-T, T]^r$ of the form $(-T + \frac{2i_1T}{n}, \cdots, -T + \frac{2i_rT}{n})$, $i_1, i_2, \cdots, i_r = 1, 2, \cdots, n$. Note that $v_n(\alpha)$ is a Riemann sum for $\int_{[-T,T]^r} \phi_\sigma(\alpha - x) f_c(x) dx$, and $\int_{[-T,T]^r} \phi_\sigma(\alpha - x) f_c(x) dx = \int_{\Re^r} \phi_\sigma(\alpha - x) f_c(x) dx = (\phi_\sigma * f_c)(\alpha)$. Thus, for any $\alpha \in \Re^r$, $v_n(\alpha) \rightarrow (\phi_\sigma * f_c)(\alpha)$ as $n \rightarrow \infty$. Since $\phi_\sigma * f_c \in L^p(\Re^r)$, there is a positive T_1 such that $\int_{\Re^r \setminus [-T_1, T_1]^r} \mid (\phi_\sigma * f_c)(\alpha) \mid^p d\alpha < (\epsilon/9)^p$. Since ϕ_σ is bounded and $\phi_\sigma \in L^1(\Re^r)$, we have $\phi_\sigma \in L^p(\Re^r)$. Thus, there exists $T_2 > 0$ such that $\int_{\Re^r \setminus [-T_2, T_2]^r} \mid \phi_\sigma(\alpha) \mid^p d\alpha < (\frac{\epsilon}{9\|f_c\|_\infty (2T)^r})^p$.

Note that $\mid v_n(\alpha) \mid \leq \parallel f_c \parallel_\infty (2T)^r \frac{1}{n^r} \sum_{i=1}^{n^r} \mid \phi_\sigma(\alpha - \alpha_i) \mid$. By Jensen's inequality [6, p.62],

$$(\frac{1}{n^r} \sum_{i=1}^{n^r} \mid \phi_\sigma(\alpha - \alpha_i) \mid)^p \leq \frac{1}{n^r} \sum_{i=1}^{n^r} \mid \phi_\sigma(\alpha - \alpha_i) \mid^p .$$

Therefore,

$$| v_n(\alpha) |^p \le [\| f_c \|_\infty (2T)^r]^p \frac{1}{n^r} \sum_{i=1}^{n^r} | \phi_\sigma(\alpha - \alpha_i) |^p .$$

Define $T_0 = \max(T_1, T_2 + T)$. Using $| \alpha_{ij} | \le T$ for all $j \in \{1, 2, \cdots, r\}$,

$$\int_{\Re^r \setminus [-T_0, T_0]^r} | \phi_\sigma(\alpha - \alpha_i) |^p \, d\alpha \le \int_{\Re^r \setminus [-T_2, T_2]^r} | \phi_\sigma(\alpha) |^p \, d\alpha,$$

and so

$$\int_{\Re^r \setminus [-T_0, T_0]^r} | v_n(\alpha) |^p \, d\alpha < (\epsilon/9)^p. \tag{3}$$

Also,

$$\int_{\Re^r \setminus [-T_0, T_0]^r} | (\phi_\sigma * f_c)(\alpha) |^p \, d\alpha < (\epsilon/9)^p, \tag{4}$$

because $T_0 \ge T_1$. Since $\phi_\sigma * f_c \in L^p(\Re^r)$ and $\| v_n \|_\infty \le \| \phi_\sigma \|_\infty \| f_c \|_\infty (2T)^r$, one has

$$\int_{[-T_0, T_0]^r} | (\phi_\sigma * f_c)(\alpha) - v_n(\alpha) |^p \, d\alpha \to 0 \text{ as } n \to \infty$$

by the dominated convergence theorem. Thus, there is an $N \in \mathcal{N}$ for which

$$\int_{[-T_0, T_0]^r} | (\phi_\sigma * f_c)(\alpha) - v_N(\alpha) |^p \, d\alpha < (\epsilon/9)^p.$$

Therefore, using (3) and (4),

$$\begin{gathered} \| v_N - \phi_\sigma * f_c \|_p \\ \le \| v_N \cdot 1_{\Re^r \setminus [-T_0, T_0]^r} \|_p + \| (v_N - \phi_\sigma * f_c) \cdot 1_{[-T_0, T_0]^r} \|_p + \| (\phi_\sigma * f_c) \cdot 1_{\Re^r \setminus [-T_0, T_0]^r} \|_p \\ < \epsilon/9 + \epsilon/9 + \epsilon/9 = \epsilon/3. \end{gathered}$$

From the above, $\| v_N - f \|_p < \epsilon$. Since

$$v_N(\cdot) = \sum_{i=1}^{N^r} \phi_\sigma(\cdot - \alpha_i) f_c(\alpha_i) (\frac{2T}{N})^r = \sum_{i=1}^{N^r} w_i K(\frac{\cdot - \alpha_i}{\sigma}) \in S_K,$$

with

$$w_i = \frac{1}{\sigma^r} f_c(\alpha_i) (\frac{2T}{N})^r \frac{1}{\int_{\Re^r} K(x) dx},$$

the proof is complete.

By K radially symmetric, we mean that $\| x \|_2 = \| y \|_2$ implies $K(x) = K(y)$. In this case, the activation function $g : [0, \infty) \to \Re$ is obtained by defining $g(d) = K(z)$, where z is any element of $\Re^r$ such that $\| z \|_2 = d$. Therefore, in the case of radial symmetry, (2) can be written as

$$q(x) = \sum_{i=1}^{M} w_i \cdot K(\frac{x - z_i}{\sigma}) = \sum_{i=1}^{M} w_i \cdot g(\frac{\| x - z_i \|_2}{\sigma}).$$

Note that there is no requirement of radial symmetry of the kernel function K in the above theorem. Thus, the theorem is stronger than necessary for RBF networks, and might be useful for other purposes. Similarly, in the following theorem and corollaries, radial symmetry of the kernel function K is not assumed, even though we are interested primarily in radial-basis-function networks.

If we interpret the term "radially symmetric" more generally than literally, then we may say that K is radially symmetric with respect to $\| \cdot \|$ if $\| x \| = \| y \|$ implies $K(x) = K(y)$, where $\| \cdot \|$ is some norm defined on $\Re^r$. With this generalization in mind, we sometimes use $\| x - z_i \|$ for the distance between x and z_i instead of $\| x - z_i \|_2$.

A slight modification of Theorem 1 given below addresses the case in which the function f we wish to approximate with an RBF network is not an element of $L^p(\Re^r)$, but an element of $L^p_{loc}(\Re^r)$ for some $p \in [1, \infty)$. Here the locally-L^p space $L^p_{loc}(\Re^r)$, $1 \leq p < \infty$ is defined as the set of all measurable $f : \Re^r \to \Re$ such that $f \cdot 1_{[-N,N]^r} \in L^p(\Re^r)$ for every $N \in \mathcal{N}$. One way to define a metric on $L^p_{loc}(\Re^r)$ is by

$$\rho_{loc}(f, g) = \sum_{n=1}^{\infty} 2^{-n} \cdot \frac{\| (f-g) \cdot 1_{[-n,n]^r} \|_p}{1 + \| (f-g) \cdot 1_{[-n,n]^r} \|_p}.$$

The following is direct corollary of Theorem 1.

Corollary 1: Let $K : \Re^r \to \Re$ be an integrable bounded function such that K is continuous almost everywhere and $\int_{\Re^r} K(x)dx \neq 0$. Then the family S_K is dense in $L^p_{loc}(\Re^r)$ for every $p \in [1, \infty)$.

Proof:

Let $p \in [1, \infty)$, $f \in L^p_{loc}(\Re^r)$, and $\epsilon > 0$. Choose $m \in \mathcal{N}$ such that $\sum_{n=m+1}^{\infty} 2^{-n} < \epsilon/2$.
Since $f \cdot 1_{[-m,m]^r} \in L^p(\Re^r)$, by Theorem 1 there is a $v \in S_K$ such that $\| f \cdot 1_{[-m,m]^r} - v \|_p < \epsilon/2$. Thus,

$$\begin{aligned} \rho_{loc}(f, v) &\leq \sum_{n=m+1}^{\infty} 2^{-n} + \sum_{n=1}^{m} 2^{-n} \| (f-v) \cdot 1_{[-n,n]^r} \|_p \\ &< \epsilon/2 + \| (f-v) \cdot 1_{[-m,m]^r} \|_p < \epsilon, \end{aligned}$$

which establishes the corollary.

Theorem 1 and Corollary 1 concern approximation with respect to the L^p metric or a metric induced by L^p metric. We next give a theorem concerning the approximation of continuous functions with respect to a metric induced by the uniform metric.

Theorem 2: Let $K : \Re^r \to \Re$ be an integrable bounded function such that K is continuous and $\int_{\Re^r} K(x)dx \neq 0$. Then the family S_K is dense in $C(\Re^r)$ with respect to the metric d defined by

$$d(f, g) = \sum_{n=1}^{\infty} 2^{-n} \frac{\| (f-g) \cdot 1_{[-n,n]^r} \|_\infty}{1 + \| (f-g) \cdot 1_{[-n,n]^r} \|_\infty}.$$

Proof:

Let $f : \Re^r \to \Re$ be any continuous function, and $\epsilon > 0$. Define $\phi : \Re^r \to \Re$ by normalizing K, and define $\phi_\sigma : \Re^r \to \Re$ for $\sigma > 0$ as in the proof of Theorem 1.

Pick a natural number m such that $2^{-m} < \epsilon/3$, and then choose a positive T such that $T > m$.

Since f is continuous on the compact set $[-m, m]^r$, we can obtain a nonzero continuous function $\tilde{f} : \Re^r \to \Re$ with the property that $\tilde{f}(x) = f(x)$ for $x \in [-m, m]^r$, and $\tilde{f}(x) = 0$ for $x \in \Re^r \setminus [-T, T]^r$. Note that $\tilde{f}$ is bounded and uniformly continuous.

Using $\phi \in L^1(\Re^r)$, pick a positive T_0 such that

$$\int_{\Re^r \setminus [-T_0, T_0]^r} \mid \phi(x) \mid dx < \frac{\epsilon}{12 \parallel \tilde{f} \parallel_\infty}. \tag{5}$$

Since $\tilde{f}$ is uniformly continuous, there is a $\delta > 0$ for which $\parallel x - y \parallel_2 < \delta$ implies

$$\mid \tilde{f}(x) - \tilde{f}(y) \mid < \frac{\epsilon}{6 \parallel \phi \parallel_1}. \tag{6}$$

Choose $\sigma > 0$ such that $\parallel \sigma x \parallel_2 < \delta$ for all $x \in [-T_0, T_0]^r$. Let $\alpha \in [-m, m]^r$. Then using (5) and (6),

$$\mid (\phi_\sigma * \tilde{f})(\alpha) - \tilde{f}(\alpha) \mid \leq \int_{\Re^r} \mid \tilde{f}(\alpha - \sigma x) - \tilde{f}(\alpha) \mid \cdot \mid \phi(x) \mid dx$$

$$\leq \int_{[-T_0, T_0]^r} \mid \tilde{f}(\alpha - \sigma x) - \tilde{f}(\alpha) \mid \cdot \mid \phi(x) \mid dx + \int_{\Re^r \setminus [-T_0, T_0]^r} 2 \parallel \tilde{f} \parallel_\infty \cdot \mid \phi(x) \mid dx < \epsilon/3. \tag{7}$$

Note that $(\phi_\sigma * \tilde{f})(\alpha) = \int_{[-T,T]^r} \phi_\sigma(\alpha - x) \tilde{f}(x) dx$. Define $v_n : \Re^r \to \Re$ by

$$v_n(\alpha) = \sum_{i=1}^{n^r} \phi_\sigma(\alpha - \alpha_i) \tilde{f}(\alpha_i) (\frac{2T}{n})^r,$$

where the set $\{\alpha_i \in \Re^r : i = 1, 2, \cdots, n^r\}$ consists of all points in $[-T, T]^r$ of the form $(-T + \frac{2 i_1 T}{n}, \cdots, -T + \frac{2 i_r T}{n})$, $i_1, \cdots, i_r = 1, \cdots, n$.

Since $\phi_\sigma(\alpha - \cdot) \tilde{f}(\cdot)$ is uniformly continuous on $[-T, T]^r$, there is a $\delta_0 > 0$ such that $x, y \in [-T, T]^r$ with $\parallel x - y \parallel_2 < \delta_0$ implies $\mid \phi_\sigma(\alpha - x) \tilde{f}(x) - \phi_\sigma(\alpha - y) \tilde{f}(y) \mid < \frac{\epsilon}{3(2T)^r}$. It easily follows that for $n > 2\sqrt{r} T / \delta_0$,

$$\mid v_n(\alpha) - \int_{[-T,T]^r} \phi_\sigma(\alpha - x) \tilde{f}(x) dx \mid < \epsilon/3. \tag{8}$$

Choose $N \in \mathcal{N}$ such that $N > 2\sqrt{r} T / \delta_0$. Then using (7) and(8),

$$\mid v_N(\alpha) - \tilde{f}(\alpha) \mid < 2\epsilon/3,$$

in which $\alpha \in [-m, m]^r$ is arbitrary. Since $f(x) = \tilde{f}(x)$ for $x \in [-m, m]^r$,

$$d(v_N, f) = \sum_{n=1}^{\infty} 2^{-n} \frac{\parallel (f - v_N) 1_{[-n,n]^r} \parallel_\infty}{1 + \parallel (f - v_N) 1_{[-n,n]^r} \parallel_\infty} \leq \parallel (\tilde{f} - v_N) 1_{[-m,m]^r} \parallel_\infty + \sum_{n=m+1}^{\infty} 2^{-n} < \epsilon,$$

which finishes the proof.

The statement in Theorem 2 is equivalent to the statement that S_K is uniformly dense on compacta in $C(\Re^r)$ under the indicated conditions on K. That is, under the conditions on K of Theorem 2, for any continuous function $f : \Re^r \rightarrow \Re$, for any $\epsilon > 0$, and for any compact subset $C \subset \Re^r$, there exists a $q \in S_K$ such that $\| (q - f) \cdot 1_C \|_\infty < \epsilon$. Thus, by a useful relationship between uniform convergence on compacta and convergence in measure [2, lemma 2.2], we have the following corollary:

Corollary 2: Let μ be a finite measure on $\Re^r$, Then under the conditions on K of Theorem 2, the family S_K is dense in $C(\Re^r)$ with respect to the metric ρ_μ defined by $\rho_\mu(f, g) = \inf\{\epsilon > 0 : \mu\{x \in \Re^r : | f(x) - g(x) | > \epsilon\} < \epsilon\}$.

3 Conclusions and Discussion

The results in Section 2 establish that under certain mild conditions on the kernel function, radial-basis-function networks having one hidden-layer and the same smoothing factor in each kernel are broad enough for universal approximation. This provides an analytical basis for the design of neural networks using radial basis functions.

To the extent that the results of this paper bear on the approximation of a function in $L^1(\Re^r)$ with a finite sum $\sum_{i=1}^{M} w_i \cdot K(\frac{\cdot - z_i}{\sigma_i})$ of kernel functions, there is some overlap of a part of [1] and this study. Using a theorem due to Wiener [7, p.210] and the pertinent argument used in [1], it can be shown that the set $\{\sum_{i=1}^{M} w_i \cdot K(\frac{\cdot - z_i}{\sigma_i}) : M \in \mathcal{N}, w_i \in \Re, z_i \in \Re^r, \sigma_i > 0\}$ is dense in $L^1(\Re^r)$, under the condition that $K \in L^1(\Re^r)$. This certainly shows the capability of certain RBF networks with respect to approximating an arbitrary L^1 function. However, note that here the smoothing factor σ_i in each kernel node has a full degree of freedom, i. e. the σ_i's can have different values across the kernel nodes. Thus, the major differences between this L^1 approximation and the results given in Section 2 concern the class of RBF networks considered as well as the metrics used.[1]

From the theoretical point of view, this condition concerning the same smoothing factor is often very important, because many studies are concerned with approximation using the functions $\sum_{i=1}^{M} w_i \cdot h(\| \cdot - z_i \|)$ ([8], [10], [11]), and radial basis functions with the same smoothing factor in each kernel node are often used in real applications ([8],[9]). In connection with studies of approximation using radial basis functions, the recent results concerning the solvability of radial-function interpolation ([10], [11]) are interesting, because they are directly applicable to the training of neural networks of the type we have focused attention on. These studies ([10], [11]) are concerned with the interpolation of data by the m functions $h(\| \cdot - z_i \|)$, $i = 1, \cdots, m$, when the data (z_i, y_i) with $z_i \in \Re^r, y_i \in \Re, i = 1, \cdots, m$ are given. More precisely, the existence of a unique interpolant $\sum_{i=1}^{m} w_i \cdot h(\| \cdot - z_i \|)$ for distinct data (z_i, y_i) with $z_i \in \Re^r, y_i \in \Re, i = 1, \cdots, m$ has been shown for a certain class of pairs of h and $\| \cdot \|$. This existence leads us an interesting observation: Suppose that traing data $(z_i, y_i), i = 1, \cdots, m$ are given, where $z_i \in \Re^r, y_i = 1$ if $z_i \in A, y_i = -1$ if $z_i \in B$, and $A, B \subseteq \Re^r$ with $A \cap B = \emptyset$. From the given data, construct a new data set

---

[1] In this connection, Wiener's theorem referred to above can also be used to give a direct proof that L^1 approximations can be achieved with linear combinations of translates of any element of $L^1(\Re^r)$ whose Fourier transform never vanishes. The Gaussians $\exp(-\alpha \| \cdot \|_2^2)$ are examples of such functions.

$z_i^* \in \Re^m, i = 1, \cdots, m$, by defining

$$z_i^* = (g(\frac{\| z_i - z_1 \|}{\sigma}), \cdots, g(\frac{\| z_i - z_m \|}{\sigma})).$$

Note that $z_i^* \in \Re^m$, while $z_i \in \Re^r$. Then by the above existence property, for certain classes of g and $\| \cdot \|$, there exist $\lambda_j \in \Re, j = 1, \cdots, m$ such that for each $i \in \{1, 2, \cdots, m\}$,

$$y_i = \sum_{j=1}^{m} \lambda_j \cdot g(\frac{\| z_i - z_j \|}{\sigma}).$$

Thus, with $\Lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)^T$, $z_i^* \Lambda > 0$ if $z_i \in A$, and $z_i^* \Lambda < 0$ if $z_i \in B$. In other words, $\{(z_i^*, y_i) : i = 1, 2, \cdots, m\}$ are linearly separable in this case. Therefore, the perceptron learning rule suffices for the training of this network.

References

[1] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Functions.", Mathematics of Control, Signals, and Systems 2, pp. 303-314, 1989.

[2] K. Hornik, M. Stinchcombe and H. White, "Multilayer Feedforward Networks Are Universal Approximators.", Neural Networks 2, pp. 359-366, 1989.

[3] R. Lippman, "Pattern Classification Using Neural Networks.", IEEE Communications Magazine, pp. 47-64, 1989.

[4] M. Stinchcombe and H. White, "Universal Approximation Using Nonsigmoid Hidden Layer Activation Functions.", Proc. of International Joint Conference on Neural Networks, pp. I613-I617, 1989.

[5] S. Bochner and K. Chandrasekharan, Fourier Transform, Princeton University Press, 1949.

[6] W. Rudin, Real and Abstract Analysis, 3rd ed., McGraw Hill, 1986.

[7] W. Rudin, Functional Analysis, McGraw Hill, 1973.

[8] D. S. Broomhead and D. Lowe, "Multi-variable Functional Interpolation and Adaptive Networks.", Complex System 2, pp. 321-355, 1988.

[9] S. Renals and R. Rohwer, "Phoneme Classification Experiments Using Radial Basis Functions.", Proc. of International Joint Conference on Neural Networks, pp. I461-I467, 1989.

[10] M. Powell, "Radial Basis Functions for Multi-variable Interpolation: a Review.", IMA Conference on Algorithms for the Approximation of Functions and Data, RMCS Shrivenham, UK, 1985.

[11] X. Sun, "On the Solvability of Radial Function Interpolation.", Approximation Theory VI: vol.2, pp. 643-646, 1989.

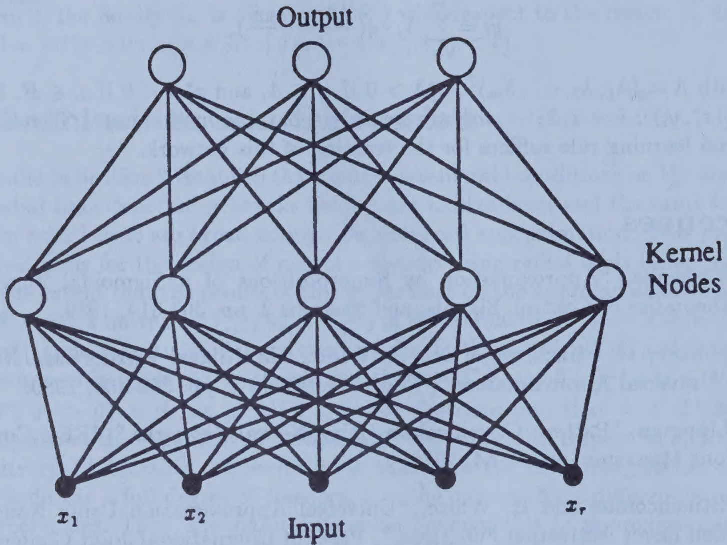

Output
Kernel
Nodes
x_1
x_2
x_r
Input

Figure 1. A radial-basis-function network.

A CONTROLLED GRADIENT SEARCH FOR THE BACK-PROPAGATION ALGORITHM*

D. T. Flaherty A. N. Michel

Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

In the multiple-layer perceptron, the algorithm used to train the neural weights is called the "Back-Propagation" algorithm. This algorithm uses a gradient search method, specifying a series of steps for calculating and propagating the network error, and determining the direction and magnitude of change in each of the neural weights. Finally, the algorithm specifies a constant adaptive gain, a global factor in the network learning rate.

This paper shows that the back-propagation algorithm *itself* may be viewed as a digital control feedback-loop system, and the neural weights, input signals, network output and error, and adaptive gain may be interpreted as variables used in digital control theory. This viewpoint enables use of many powerful concepts and results from control theory to improve back-propagation. The paper also discusses methods of reducing variables in the gradient search, such as fixing the threshold.

1. INTRODUCTION

For the past few years, a great deal of attention has been given to modeling, analysis procedures, and applications of artificial neural networks[5-7,9-12]. This research complements interest in the field of neurophysiology in its goal of providing a better understanding of massively interconnected neural structures. The artificial models that have been studied are numerous; one area of which concerns networks which use supervised learning to train these neural structures to associate ranges of analog input signals with desired classifications. The "Feed-forward" or "Perceptron" arrangement of neurons using this supervised learning is the topic of this paper.

Research in the multiple-layer perceptron arrangement typically utilizes the supervised algorithm called "Back-propagation" to alter the neural weights. This algorithm uses a method of gradient descent toward a composition of the neural weights which best associates the analog input signals with the desired classifications. A large variety of methods have been developed to study the convergence of this descent in terms of energy functions and Lyapunov stability tests [1], comprising a viewpoint of the behavior of the network as it attempts to learn the data. In order to supplement this "energy-descent" viewpoint, a new model of the same algorithm has been established, enabling use of powerful digital-control feedback theorems in the convergence study.

Section 1.1 defines the variables and arrangement typical of the literature for a single neuron. Section 1.2 defines the arrangement and variables commonly used in a simple discrete-control feedback loop for a linear single-input, single-output system. The association between these two models is discussed in section 2.1, and section 2.2 discusses the changes necessary for modeling the algorithm for the multi-neuron, multi-layer perceptron.

---

* supported in part by the National Science Foundation under Grant ECS 88-02924.

1.1. Definitions of Perceptron Variables

Figure 1.1 depicts a single-neuron model, with signal input vector $\Gamma(t) = (\gamma_1(t), \gamma_2(t), \ldots, \gamma_n(t))^T$ at iteration t, neural weight vector $W(t) = (w_1(t), w_2(t), \ldots, w_n(t))^T$, threshold $\theta(t)$, and output $y(t)$. The output is defined as

$$y(t) = f\left(\Gamma(t)^T W(t)\right) \tag{1.1}$$

where $f(x) \equiv \frac{2}{\pi} \tan^{-1} \left\{ \frac{\pi}{2}(x - \theta(t)) \right\}$. This output, $y(t)$, is compared with the desired output $d(t)$ for the given input $\Gamma(t)$, and the difference is the output error, $\delta(t)$, of the neuron for that iteration. This error is then multiplied by the adaptive gain, η, to determine how far the weights are to be moved:

$$W(t+1) = W(t) + \eta\, \delta(t)\, \Gamma(t). \tag{1.2}$$

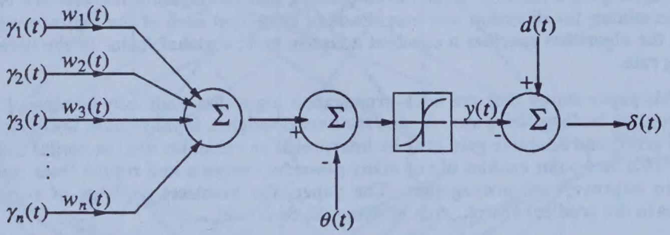


Figure 1.1: Single Neuron in Perceptron Variables

1.2. Definitions of Discrete Control Variables

In order to describe a discrete-time, single-input, single-output system with $n \times 1$ state vector $X(k)$ for iteration k, input $u(k)$, and output $y(k)$, the $n \times n$ matrix $\Phi(k)$, the $n \times 1$ vector $\Gamma(k)$, and the $1 \times n$ vector $C(k)$ can be defined in the following state and output equations [2]:

$$X(k+1) = \Phi(k)\, X(k) + \Gamma(k)\, u(k) \tag{1.3}$$

$$y(k) = C(k)\, X(k). \tag{1.4}$$

To describe a control-feedback loop for this system, an arbitrary controller can be expressed as having pulse transfer characteristics $G(q)$, where q is the discrete-time shift operator [3]. Furthermore, letting $r(k)$ be the desired output of the system for iteration k, and $e(k) \equiv r(k) - y(k)$ the error, the system input $u(k)$ can be expressed as

$$u(k) \equiv G(q)\, e(k) \tag{1.5}$$

the control law. Figure 1.2 illustrates this.

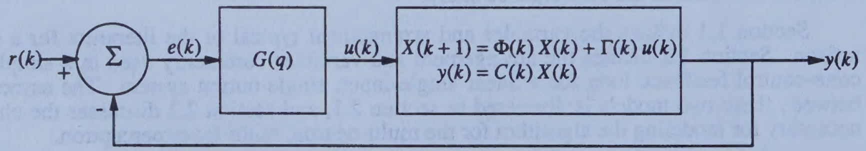


Figure 1.2: Discrete Control Feedback Loop

It should be noted that both Figure 1.1 and Figure 1.2 depict systems with states, outputs, desired outputs, and errors. The next section shows how the variables of section 1.2 can be defined in order to associate the two.

2. FEEDBACK-LOOP VIEWPOINT

As mentioned in the introduction, the feedback loop of Figure 1.2 can be used to describe the back-propagation algorithm shown for one neuron in Figure 1.1. This association is made by defining the variables in section 1.2 in terms of those in section 1.1.

2.1. Association of Variables in Equations

Letting k replace t as the iteration index, the following variable assignments are made: to associate equation (1.2) with the state equation (1.3), fix $\Phi(k)$ to I_n the $n \times n$ identity matrix, replace the product $\eta\,\delta(t)$ with the system input $u(k)$, and let $\Gamma(k) \equiv \Gamma(t)$. The weight vector $W(t)$ in the neural model is associated with the state of the discrete system, $X(k)$. For the output equation (1.1), let the system output $y(k)$ replace $y(t)$ and $C(k) \equiv \Gamma^T(k)$. This requires that the output equation of (1.4) is expressed as a nonlinear function,

$$y(k) = f\big(C(k)\,X(k)\big) \tag{2.1}$$

where $f(x)$ is now $\frac{2}{\pi}\tan^{-1}\left\{\frac{\pi}{2}(x - \theta(k))\right\}$. Equations (1.3) and (2.1) then represent the state and output equations of the back-propagation algorithm. Finally, to complete the loop, the desired output $r(k)$ replaces $d(t)$, and the error $e(k)$ replaces $\delta(t)$. Using the control law of equation (1.5), this requires that the constant adaptive gain η is associated with the controller $G(q)$. Table 2.1 lists this association in its entirety.

Perceptron Figure 1.1	*variables*	Feedback Loop Figure 1.2
t	discrete time/iteration	k
I	identity	$\Phi(k)$
$\eta\,\delta(t)$	system input	$u(k)$
$\Gamma(t)$	neural input	$\Gamma(k)$
$W(t)$	neural weights	$X(k)$
$y(t)$	neural output	$y(k)$
$d(t)$	desired output	$r(k)$
$\delta(t)$	output error	$e(k)$
η	adaptive gain	$G(q)$

Table 2.1

Use of a controller, $G(q)$, that is more general than a constant η provides much greater flexibility in the update process. In addition, any variety of modern control designs utilizing feedforward paths, prefilters, or model followers may be employed to further shape the algorithm [4]. In section 3, the popular "PID" controller is used as an example for simplicity.

2.2. Expansion into Multi-neuron, Multi-layer Network

Since the applications for a perceptron with only one neuron are limited, the association of a perceptron with a discrete-control feedback loop must be expanded to the multi-neuron, multi-layer case. First, the expansion to m neurons within a single (output) layer with n input signals is accomplished by a horizontal expansion of the variables $u(k)$, $y(k)$, $r(k)$, and $e(k)$ into m-length row vectors $U(k)$, $Y(k)$, $R(k)$, and $E(k)$, where each member of a vector corresponds to the system input, neural output, desired output, or output error, respectively, of a neuron. The state $X(k)$ is now an $n \times m$ matrix, constructed by concatenating m columns, one for each neuron, of $n \times 1$ weight vectors. $\Gamma(k)$ remains the $n \times 1$ neural input vector, making $\Gamma(k)\,U(k)$ an $n \times m$ outer product that matches the dimensions of $X(k)$. The matrix function $f(x)$ is defined element-wise. This single, output layer is depicted by Figure 2.1.

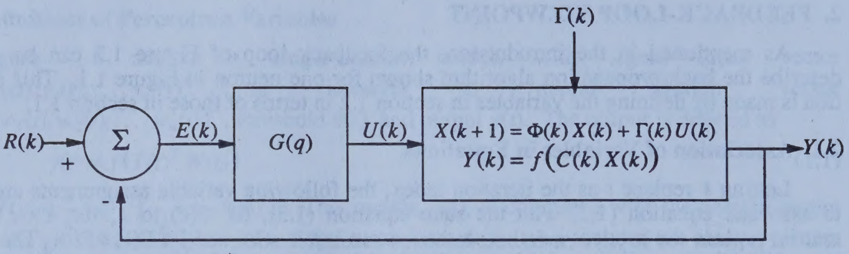


Figure 2.1: Feedback Loop for Output Layer

Second, if a hidden layer with n neurons is to precede this output layer, the path of Figure 2.2 must be prepended above the loop in Figure 2.1. It is attached by placing the starting point at the error $E(k)$ in the output layer, and the path concludes by sending its output $Y_h(k)$ to the neural input $\Gamma(k)$ of the output layer. The hidden-layer error vector $E_h(k)$ is thus defined in terms of the error vector and state matrix of the output layer:

$$E_h(k) \equiv E(k)\, X^T(k) \tag{2.2}$$

and the neural input $\Gamma(k)$ of the output layer is simply

$$\Gamma(k) \equiv Y_h^T(k). \tag{2.3}$$

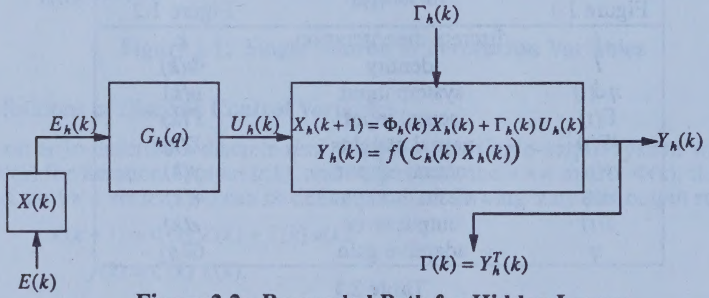

Γ_h(k)
E_h(k)
G_h(q)
U_h(k)
X_h(k + 1) = Φ_h(k) X_h(k) + Γ_h(k) U_h(k)
Y_h(k) = f(C_h(k) X_h(k))
Y_h(k)
X(k)
E(k)
Γ(k) = Y_h^T(k)

Figure 2.2: Prepended Path for Hidden Layers

Further layers may be prepended in a similar manner. A system is defined for each layer, where the vectors for system input, system output, and output error are row vectors having lengths equal to the number of neurons in that layer. The neural input vector and the state matrix always have as many rows as the number of neurons in the preceding layer, if there is one; or the number of input signals, if not.

If the controller $G(q)$ is a scalar, the control law

$$U(k) \equiv G(q)\, E(k) \tag{2.4}$$

is sufficient. However, a square matrix may be used for $G(q)$, in which case transposed multiplication must be used,

$$U(k) \equiv \left[G(q)^T\, E(k)^T\right]^T \tag{2.5}$$

for matrix multiplication to be defined. A matrix $G(q)$ with gains along the diagonal allows for different gains for individual neurons within a layer, and if off-diagonal terms are used, the error of one neuron can affect the update of another. The examples of section 3 use equation (2.4), scalar $G(q)$.

2.3. Revised Update Equation in Perceptron Variables

The change in variables presented in this paper from those of section 1.1 to those of section 1.2 is intended to make the association of back-propagation with the feedback-loop viewpoint more clear. It is not intended to further confound efforts to standardize the emerging field of neural networks as to what variables are commonly used. This section discusses the compatibility of the results of this paper with other current developments.

Section 2.1 discussed the change of variables necessary for the feedback loop of Figure 1.2 to model the single-neuron perceptron of Figure 1.1. The association of the adaptive gain η with a controller $G(q)$ makes it possible for more general adaptive gain characteristics. Although the proposed viewpoint of the back-propagation algorithm is the central purpose of this paper, the results that can be used in other works stem from the greater flexibility allowed by using general pulse transfer characteristics for a learning rate. Therefore, the update equation (1.2) for the algorithm can be rewritten as

$$W(t+1) = W(t) + G(q)\,\delta(t)\,\Gamma(t). \tag{2.6}$$

Use of $G(q)$ is compatible with other developments in back propagation, such as improving the speed of convergence using guard bands [5], or by improving the likelyhood of convergence using neuron clustering [6]. The controller even generalizes "weight decay" or "momentum" factors as differential gain, a relationship that will be discussed in the section on differential gain, section 3.2. In general, results from section 3 can be used to improve convergence of back propagation by usage of equation (2.6) as an update equation.

3. FEATURES OF THE CONTROLLER

With the discrete-control feedback loop viewpoint of the back-propagation algorithm established, the discussion can now be focussed on the controller, $G(q)$. Although many more control concepts can be used to shape the convergence of the algorithm, the simplest is the popular "PID" controller, which uses proportional, integral, and differential gain terms. This section discusses how these terms can be used, and what results can be expected.

3.1. Description of Examples

Two examples are used throughout this section. The first is the degenerate case of a single neuron, pictured by Figure 1.1, whose feedback loop is described by Figure 1.2 with the state and output equations (1.3) and (2.1). This neuron will have four weights, and thus four input signals. According to the previous discussion, the state equation (1.3) will be

$$X(k+1) = \Phi(k)\,X(k) + \Gamma(k)\,U(k)$$

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} + \begin{bmatrix} \gamma_1(k) \\ \gamma_2(k) \\ \gamma_3(k) \\ \gamma_4(k) \end{bmatrix} \begin{bmatrix} u(k) \end{bmatrix} = \begin{bmatrix} x_1(k)+\gamma_1(k)u(k) \\ x_2(k)+\gamma_2(k)u(k) \\ x_3(k)+\gamma_3(k)u(k) \\ x_4(k)+\gamma_4(k)u(k) \end{bmatrix}$$

and the output equation (2.1) would be

$$Y(k) = f\big(C(k)\,X(k)\big) = f\big(\Gamma^T(k)\,X(k)\big)$$

$$\begin{bmatrix} y(k) \end{bmatrix} = f\left(\begin{bmatrix} \gamma_1(k) & \gamma_2(k) & \gamma_3(k) & \gamma_4(k) \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} \right)$$

$$= f\Big(\Big[\gamma_1(k)x_1(k)+\gamma_2(k)x_2(k)+\gamma_3(k)x_2(k)+\gamma_4(k)x_4(k)\Big]\Big).$$

The controller is a scalar $G(q)$ expanded as

$$G(q) = \mu_P + \frac{q}{q-f}\mu_I + \frac{q-1}{q}\mu_D \tag{3.1}$$

where μ_P is the proportional gain (which directly replaces η used before), μ_I is the integral gain, f is the integrator forgetting factor, and μ_D is the differential gain.

This neuron is trained by letting $\Gamma(k)$ be a 4×1 random vector, each member of which is uniformly distributed on the interval [-1,1]. $r(k)$ is calculated by multiplying this neural input vector by a desired set of weights, and passing it through the nonlinearity, similar to equation (2.1):

$$r(k) = f\left(\begin{bmatrix}\gamma_1(k) & \gamma_2(k) & \gamma_3(k) & \gamma_4(k)\end{bmatrix}\begin{bmatrix}0.7\\0.2\\-0.3\\-0.5\end{bmatrix}\right). \tag{3.2}$$

The second example, a multi-neuron, multi-layer network, is based on the well-known example presented in Lippman's survey paper [7]. The network has two layers, with eight neurons in the first (hidden) layer and one in the second (output) layer. The output layer has the state and output equations of the single-neuron example above, except that there are eight weights and signals instead of four. The hidden layer, however, has eight neurons and two input signals, so $X_h(k)$ would be a 2×8 state matrix, $\Gamma_h(k)$ would be 2×1, and $U_h(k)$ and $Y_h(k)$ would be 1×8. The analysis of section 2.2 indicates that there are two controllers, one for each layer. However, the same scalar controller is used for both, the same expansion as for the single-neuron example, equation (3.1).

This two-layer network is a *classifier:* a circle centered at the origin with radius 0.5 is drawn in the two-input (γ_1, γ_2) domain of the network, and its goal is to classify points inside the circle as a +1, and outside as -1. Although a perfect circle cannot be separated with a finite number of hidden neurons, eight first-layer neurons can approximate a circle best by forming an octagon. Section 4 explains the details of this separation used in the simulation.

The following sections demonstrate the effects of adding differential and integral gain for both examples, by fixing the proportional gain and increasing the others.

3.2. Effects of Differential Gain

The "servo problem" in control theory is usually the term used for systems with a controller and a feedback loop, where a reference input directs the system to a desired state. To increase the speed of this action, differential gain is often added to the proportional gain [8].

In the back-propagation case, higher gains are desirable in the early training phase, but when the network has converged, lower gains are desirable. The effect of adding differential gain is to make the network learn faster initially because of the large *changes* in the error. In the steady state, if errors are small, the changes in the errors are also small, and thus weights move slowly, as desired. The graphs show these results by fixing the proportional gain, and showing the effects of differential gain as it is increased over an interval from 0.

Figure 3.1 is a graph of the four weights of the single-neuron example as they change over the iterations. The four straight, horizontal lines are placed at 0.7, 0.2, –0.3, and –0.5, the four "desired" weights in the definition of $r(k)$ in equation (3.2). At 250, the desired-weight vector is changed to $[0\ \ 0\ \ 0\ \ 0]^T$ in order to show a falling transient. For each weight, two curves are superimposed: one with only proportional gain, just enough to converge in 250 iterations, and one with the same proportional gain but added differential gain. The clear difference is how much faster the four weight curves rise and fall. Once the curves with differential gain converge, there is insignificant further variance— variance that would be much greater if proportional gain was simply increased.

When this concept is used for multilayer perceptrons, the effect of faster and cleaner convergence is shown by Figure 3.2. Here the network is trained for 10,000 iterations, and at every 2000 iterations the network is sampled to show the mean squared-error. The two curves show the comparison of with versus without differential gain, with the same proportional gain. The curve with proportional gain converges faster and shows less steady-state variance than would a curve with simply more proportional gain.

3.2.1. Comparison with "Weight Decay"

Although the concept of the μ_D term as a differential gain in a controller is presented in this paper, use of the term itself is not new. This section describes how the differential term is equivalent to "momentum" or "weight decay" terms in other papers [9], and that the feedback-control viewpoint explains features of this term that the energy-descent viewpoint does not.

The momentum term α that is often added to the gain η can be expressed as follows:

$$W(t+1) = W(t) + \left(\eta\, \delta(t) + \alpha \frac{d}{dt}\, \delta(t) \right) \Gamma(t). \tag{3.4}$$

The shift-operator expansion used for differential gain is merely a backward-rectangular discrete approximation,

$$\left(\mu_P + \mu_D \frac{q-1}{q} \right) e(k). \tag{3.5}$$

This expansion is evaluated in the simulations as the "current error" minus the "previous error" divided by the number of iterations since the previous error, which is always 1. If the iterations were to be thought of as a measure of time rather than data trials, then as the time between iterations would approach zero, the approximation in equation (3.5) would approach the derivative in equation (3.4) as a limit. Thus the terms are equivalent, even though their origins are different approaches to the gradient search.

A common reason for the back-propagation algorithm not converging is the existence of local minima in the energy surface of the neural weight space. The weights move down the energy surface in a gradient descent, but may get caught in a local minimum. For this reason, "momentum" was added onto the weights, increasing the likelihood that the minima would be passed. For more information on second-order terms, see [10–12]

The fact that this term also dramatically speeds up convergence, however, even in examples without local minima such as the single-neuron in this section, is not explained by the energy-descent viewpoint. Rather, it can be seen as the effects of differential gain in the feedback-control viewpoint. For this reason, the proposed viewpoint complements the existing one by providing explanation for features already encountered. In the next section, it will be seen that this viewpoint makes a new term obvious.

3.3. Effects of Integral Gain

The data for training upto this point can be described as "stationary". That is, the classifications that the neural network is trying to learn do not change over the iterations. In the single-neuron example, the desired weights are constant, and in the two-layer example, the circle has constant position and size. However, if these classifications change, the gradient search follows a moving energy surface, tracking the bottom of this surface with a steady-state error. Using the feedback-control viewpoint, it can be seen that this problem is similar to steady-state ramp error caused by a changing command input $r(k)$. The solution to this error is to increase the "type" of the controller, by adding integral gain [13].

Unlike the differential gain term which was used to improve the transient before convergence, integral gain is useful for maintaining convergence once it has been attained, in the steady state. The single-neuron example previously defined is used to once again show these effects.

Figure 3.3 depicts the movements of the four weights of the single neuron over the iterations, as Figure 3.1 did. This time, the desired weights in the definition of $r(k)$ are increasing linearly with time. The four straight, sloped lines are the desired weights, which begin at the same four constants, 0.7, 0.2, –0.3, and –0.5, and increase by 0.001 each per iteration. But when only proportional gain is used, each weight has a steady-state ramp error, depending on its distance from 0 (due once again to the nonlinearity).

To solve this problem of steady-state ramp error, integral gain is added. Also, differential gain is added to improve the transient. It can be readily noticed that the weights converge quickly on the nonstationary targets, and remain on the moving targets throughout the simulation with little noise.

A similar comparison is made using the multilayer perceptron used in the previous section. To make the classification boundary change in a simple way, the radius of the circle is made to decrease from a starting radius of 0.6 to a final radius of 0.12 over the iterations. To keep up with this change, all of the neural weights must move proportionally farther away from zero. Although the single-neuron example had the weights only moving in a positive direction, the concept of chasing moving weight targets is similar.

Figure 3.4 makes this comparison using the same "convergence curves" as Figure 3.2 used. The emphasis is on how much closer the curve with integral gain is to zero in the steady state.

4. VARIABLE REDUCTION

In order to simulate the examples of the previous section, a C-language computer software package was written. This section discusses an algorithm concept that was implemented in the program in order to improve the convergence properties of back-propagation.

4.1. Reduction in the First Layer

A single neuron in the first layer of a perceptron having n weights is responsible for dividing up the n-dimensional network input domain into two halves, along a hyperplane described by

$$\gamma_1(k)x_1(k) + \gamma_2(k)x_2(k) + \gamma_3(k)x_3(k) + \cdots + \gamma_n(k)x_n(k) = \theta(k) \tag{4.1}$$

at iteration k, where $x_i(k)$ is the weight for the input signal $\gamma_i(k)$ in the domain for all $i = 1, \ldots, n$, and $\theta(k)$ is the neural threshold. To uniquely define this hyperplane requires n variables, but equation (4.1) indicates that the neuron has $n+1$ variables. In particular, for *every* fixed θ, there exists a unique set of solutions for $x_1, x_2, x_3, \ldots, x_n$ such that

$$\gamma_1 x_1 + \gamma_2 x_2 + \gamma_3 x_3 + \cdots + \gamma_n x_n = \theta$$

describes the same hyperplane in the n-dimensional γ space. In addition, the hyperplane will pass through the origin if and only if $\theta = 0$.

The result of allowing the threshold of a neuron in the first layer to vary according to the back-propagation algorithm is that the weights must constantly change in proportion to the threshold in order to maintain the same hyperplane. To eliminate this redundancy, the threshold was fixed to *unity* for the experiments.

4.2. Fixed Threshold in the Entire Network

This redundancy of variables in the algorithm is evident in other layers. In general, the gradient search must search for an ideal composition of neural weights by moving in the direction of the negative gradient in weight space. Because there is a unique set of solutions for the weights of a neuron for each value of the threshold, the global minimum for the error which the algorithm searches for is also distinct for *each* set of neural thresholds. If the thresholds in the network are allowed to vary, the back-propagation algorithm is forced to search for a global minimum that is constantly changing greatly for every iteration.

Reducing the redundancy in the network by fixing the thresholds greatly increases the chances of convergence because the global minimum is more consistent. Other methods may be found for further reductions.

4.3. Reductions Based Upon Boundary Knowledge

Implementations of reduction methods are often limited to one's foreknowledge of the classification boundaries. If no boundaries pass through the origin, then all thresholds may be fixed at unity. However, if a boundary passes through the origin, then the weights of at least one neuron will increase without bound in an attempt to render the threshold negligible. For this reason, it may be necessary to add one or two neurons onto the first hidden layer with *null* thresholds— increasing the redundancy of the network— but fitting the required boundary. Experiments have shown that if null-threshold neurons are used but no boundary passes through the origin, the back-propagation algorithm reduces the corresponding second-layer weights to zero to ignore them.

CONCLUSION

This paper presents a "feedback-loop" viewpoint of the back-propagation algorithm to complement the existing "energy-descent" viewpoint commonly used to study convergence. As a result of using more general shift-operator transfer characteristics $G(q)$ as a learning rate, certain terms become obvious: differential gain improves convergence speed, while integral gain reduces steady-state ramp error. Finally, methods to reduce redundancy in the network are presented.

REFERENCES

[1] Miller, Richard K. and Michel, Anthony N., *Ordinary Differential Equations*. Acedemic Press, 1982.

[2] Åström, Karl J. and Wittenmark, Björn, *Computer-Controlled Systems*. NJ: Prentice-Hall, 2nd ed, 1990, p44.

[3] Åström, Karl J. and Wittenmark, Björn, *Computer-Controlled Systems*. NJ: Prentice-Hall, 2nd ed, 1990, p58.

[4] Kuo, Benjamin C., *Automatic Control Systems*. NJ: Prentice-Hall, 5th ed, 1987, p466.

[5] Nguyen, Dziem D. and Lee, James S. J., "A New LMS-Based Algorithm for Rapid Adaptive Classification in Dynamic Environments," *Neural Networks*, Vol. 2, No. 3, Pergamon Press, 1989, pp215-28.

[6] Gallant, Stephen I., "Perceptron-Based Learning Algorithms," *IEEE Transactions on Neural Networks*, Vol. 1, No. 2, June 1990.

[7] Lippman, Richard P., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, April 1987.

[8] Kuo, Benjamin C., *Automatic Control Systems*. NJ: Prentice-Hall, 5th ed, 1987, p468.

[9] Hanson, Stephen J. and Pratt, Lorien Y., "Comparing Biases for Minimal Network Construction with Back-Propagation," *Advances in Neural Information Processing Systems*, 1989, pp179-80.

[10] Becker, S. and le Cun, Y., "Improving The Convergence of Back-Propagation Learning With Second-Order Methods," *Proceedings of the 1988 Connectionist Models Summer School*.

[11] Chan, L. W. and Fallside, F., "An Adaptive Training Algorithm for Back-Propagation Networks," *Computer Speech and Language*, 1987, 2, p205-218.

[12] Jacobs, R. A., "Increased Rates of Convergence Through Learning Rate Adaptation," *Neural Networks*, vol 1, 1988 p295-307.

[13] Kuo, Benjamin C., *Automatic Control Systems*. NJ: Prentice-Hall, 5th ed, 1987, p475.

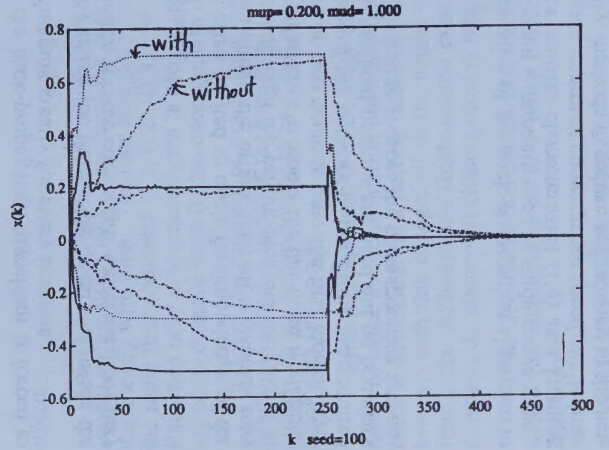


Figure 3.1: Single Neuron Differential Gain Comparison

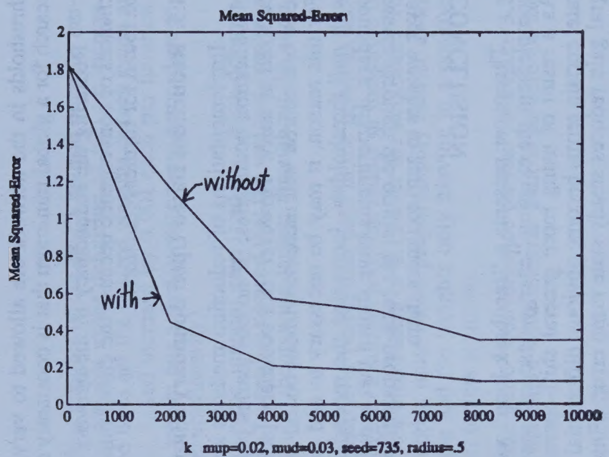


Figure 3.2: Multilayer Differential Gain Comparison

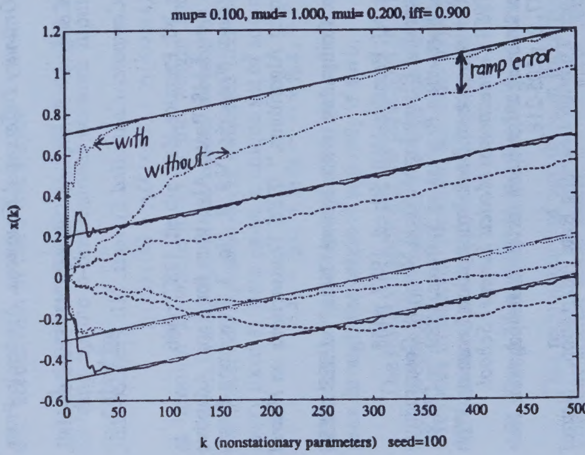


Figure 3.3: Single Neuron Integral Gain Comparison

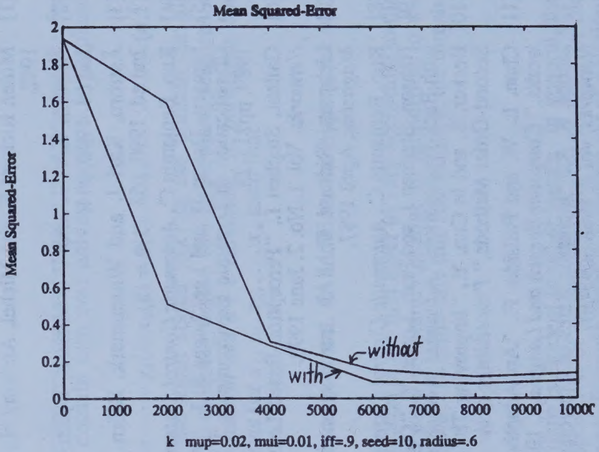


Figure 3.4: Multilayer Integral Gain Comparison

ANALYSIS AND SYNTHESIS OF A CLASS OF DISCRETE-TIME NEURAL NETWORKS WITH STAIRCASE NONLINEARITIES*

J. Si and A. N. Michel
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

In contrast to the usual types of neural networks which utilize two states for each neuron, a class of synchronous discrete-time multi-level threshold neural networks is developed. A qualitative analysis and a synthesis procedure of the calss of neural networks considered herein constitute the principal contributions of this paper. The applicability of the present class of neural networks is demonstrated by means of a grey level image processing example, where each neuron of the present model assumes one of sixteen values. In doing so, the number of neurons and the number of interconnections are reduced when compared to the usual binary state networks (which is a major concern in VLSI implementation techniques).

I. ANALYSIS

We will consider neural networks described by equations of the form

$$x_i(k+1)=\Sigma_{j=1}^{n} T_{ij}s_j(x_j(k))+I_i, \quad i=1,\ldots, \quad k=0,1,2,\ldots. \qquad \text{(S)}$$

where $x=(x_1,\ldots,x_n)^T \in R^n, T_{ij} \in R$, $I_i \in R$ and s_j: $R \rightarrow R$ is a threshold function with input-output characteristics depicted in Fig.1.

We will employ the following notation.

Let $L_m \in Z$ and $L_M \in Z$, $L_m < L_M$, denote the lowest and highest integer values of each neuron output, respectively. Then $L=L_M-L_m+1$ will be the total number of stages or possible number of states a neuron can assume.

Define $B^n=\{x \in R^n: x_i \in \{L_m,L_{m+1},\ldots,L_M\}, i=1,\ldots,n\}$.

If L is a b-bit word which is to be stored in a binary neural network, then an n×b order system may be used to store an n-vector with each component of b-bit length. Alternatively, an n-dimensional neural network may be employed for this purpose, provided that each neuron can represent a b-bit word. In the former case, the number of interconnections will be $(n \times b)^2$ while in the latter case, the number of interconnections will only be n^2.

In the following, we perform a qualitative analysis of system (S). As will be seen later, this asymptotic stability analysis is independent of the number of stages and the precise shape of the input-output characteristic function, as long as s_i: $R \rightarrow R$ is a monotonically nondecreasing continuous function with $s_i(0)=0$.

To circumvent difficulties encountered in the analysis of complex systems, such as system (S), we will find it convenient to utilize the method of analysis proposed in [15]. In this approach we view system (S) as an interconnection of free or isolated subsystems.

The structure of the remainder of this section is as follows. In the first subsection of the present section, we transform system (S) into a form necessary for a stability analysis of an equilibrium. In the second subsection, we phrase the system in a form consistent with the large scale systems approach. In the third subsection we establish stability results of the equilibrium points of system (S). In the fourth subsection of the present section, neural networks with quantization nonlinearities for the neuron models (Fig.2), are studied. Finally, neural networks endowed with neurons which are described by multi-level piecewise continuous linear saturation functions (Fig.3) are investigated.

1.1. The translation of an equilibrium of (S).

Letting $T=[T_{ij}] \in R^{n \times n}$, $I=[I_i] \in R^n$, and $s=[s_1,\ldots,s_n]^T$: $R^n \rightarrow R^n$, the neural network (S) may be described by the eqation

$$x(k+1)=Ts(x(k))+I, \qquad k=0,1,2,\ldots. \qquad \text{(S)}$$

---

*Supported in part by the National science Foundation under grant ECS 88-02924

Assumption (A-1). There are two positive real numbers c_{i1} and c_{i2} such that $c_{i1} \le s_i(x)/x \le c_{i2}$ for all $x \in R$. (E.g., for the staircase nonlinearities depicted in Fig.3, we have $c_{i1}=0$, $c_{i2} \ge 3$).

In general system (S) will have equilibria which may not be located at the origin ($x_i=0$, $i=1,\dots,n$). The equilibria of (S) may be desired memories which we have stored, or they may be extraneous equilibria. Each equilibrium must, by definition, satisfy the equation

$$x(k)=Ts(x(k))+I.$$

This equation is equivalent to

$$0=Ts(x(k))-Ex(k)+I \tag{1.1}$$

where E is the n×n identity matrix. We are able to translate any equilibrium of (S) to the origin in the following manner. Let

$$p(k)=x(k)-x^* \tag{T1}$$

$$S(p(k))=s(x(k))-s(x^*) \tag{T2}$$

where x^* satisfies equation (1.1), and $S=[S_1,\dots,S_n]^T$. Then by calculation,

$$p(k+1)=TS(p(k)) \tag{Σ}$$

which has an equilibrium at $p(k)=0$. Henceforce, we will use (Σ) to represent the model of the neural network under study.

Reamark 1.1. Due to the relationship between $S_i(x)$ and $s_i(x)$, $S_i(x)$ is also a monotonically nondecreasing, continuous function with $S_i(0)=0$. Again $S_i(x)$ will satisfy a sector condition. For example, for the nonlinearity in Fig. 3, we have $c_{i1} \le S_i(x)/x \le c_{i2}$ for all $x \in R$ (the c_{i1} and c_{i2} are given in A-1).

1.2. Interconnected systems.

We express (Σ) as

$$p_i(k+1)=\sum_{j=1}^{n} T_{ij}S_j(p_j(k)) \qquad \text{for } i=1,\dots,n, \quad k=0,1,2,\dots \tag{Σ_i}$$

System (Σ_i) may be viewed as an interconnection of n free (isolated) systems represented by the equtions

$$x_i(k+1)=T_{ii}S_i(x_i(k)) \qquad \text{for } i=1,\dots,n, \quad k=0,1,2,\dots \tag{S_i}$$

with the intercooneciing structure defined by

$$h_i(x_1,\dots,x_n) \overset{\Delta}{=} \sum_{j=1\, j\neq i}^{n} T_{ij}S(x_j(k)) \qquad \text{for } i=1,\dots,n, \quad k=0,1,2,\dots \tag{1.2}$$

Following the method of analysis in [15], we will establish stability results for system (Σ) which are phrased in terms of the qualitative properties of the free subsystems (S_i) and in terms of the properties of the interconnecting structure given in (1.2).

1.3. Stability analysis of an equilibrium point.

The stability properties of the entire neural network (Σ) are determined by the stability properties of the individual free neurons (S_i) and the properties of the interconnection structure (1.2) of the neural network. In this section, we first analyze the stability of the free subsystems. Next we establish stability results for the entire neural network (Σ) which are phrased in terms of the stability properties of the individual free subsystems (S_i) and in terms of the properties of the interconnecting structure given by (1.2).

1.3.1. Stability results of the free subsystems.

To analyze the stability of a free subsystem (S_i), we need the following assumption:

Assumption (A-2).

For (S_i),

$$0 \le \sigma_i \overset{\Delta}{=} (|T_{ii}|c_{i2}) < 1$$

where c_{i1} and c_{i2} are defined in Assumption (A-1).

Proposition 1.1.

If (A-1) and (A-2) are true, then the equilibrium $x_i=0$ of (S_i) is asymptotically stable.

The proof of this result as well as all of the remaining results are omitted due to space limitations.

1.3.2. Stability results of the neural network.

Assumption (A-3).

Given σ_i of (A-2), the successive principal minors of matrix $D=[D_{ij}]$ are all positive, with

$$Dij=\begin{cases} -(\sigma_i-1) & i=j \\ -\sigma_{ij} & i\neq j \end{cases}$$

where $\sigma_{ij}=|T_{ij}|c_{i2}$ (c_{i2} is defined in (A-1)).

Theorem 1.1. If (A-1), (A-2) and (A-3) are true, then the equilibrium p=0 of (Σ) is asymptotically stable.

Remark 1.2. If the desired equilibrium points are located on the "plateaus" of the nonlinearities (e.g., point p* in Fig.4), then c_{i2} will be very small for i=1,...,n. In this case (A-2) and (A-3) are satisfied and the asymptotic stability of the desired equilibrium points is guaranteed.

1.3.3. Global stability results.

A design by the method presented in [11] will in general not result in a symmetric interconnection matrix T and will in general not result in a neural network which is globally stable. In the present paper we modify the design procedure of [11] to yield neural networks with symmetric interconnection structure. We will show that this resulting network, with neuron models specified, e.g., by Fig.1 or Fig.2, are globally stable. In the case of networks which utilize neuron models described, e.g., by Fig.3, we are unable (at this time) to establish global stability for neural networks designed by the method advanced herein.

To establish global stability for (S) under appropriate assumptions, we will make use of an energy function of the form

$$E(x(k))=-\frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}T_{ij}v_i(k)v_j(k)-\Sigma_{i=1}^{n}v_i(k)I_i+\Sigma_{i=1}^{n}\int_0^{v_i}s_i^{-1}(\sigma)d\sigma \tag{1.3}$$

where $v_i(k)=s_i(x_i(k))$, i=1,...,n.

Theorem 1.2. Suppose the interconnection matrix T in (S) is positive semidefinte and the input-output nonlinearities $s_i(\cdot)$ are strictly increasing. Then along non-equilibrium solutions of system (S), the energy function given in (1.3) decreases monotonically, and each non-equilibrium solution of (S) converges to an equilibrium point of system (S).

1.4. Systems with quantization nonlinearities.

Global convergence of system (S) with the input-output characteristics of the neurons specified as quantization nonlinearities is preserved. We will address this issue in the present subsection.

The class of neural networks (S) with neurons characterized by the signum nonlinearity, and operating asynchronously in a random order, will be denoted as system (D). Hopfield [1]-[3] has shown that the global stability of system (D) is guaranteed when $T_{ij}=T_{ji}$, and $T_{ii}=0$ by using an energy function $E(x)=-\frac{1}{2}x^TTx-x^TI$. Furthermore, he has shown that this energy function will decrease along the evolution of the network, and that eventually, the network will tend to a local minimum of E(x). In [13] and [14], the authors extended the above results by showing that the asynchronous operation of system (D) will maintain global stability or only has fixed points if the following conditions are satisfied: (1) $T_{ij}=T_{ji}$ and (2) T_{ii} is nonnegtive. Furthermore , by using an energy function [14], $E(x(k))=-\frac{1}{2}\Sigma_{i=1}^{n}x_i(k)\Sigma_{j=1}^{n}T_{ij}x_j(k-1)-\Sigma_{i=1}^{n}x_i(k)I_i$, the authors proved that for a general symmetric interconnection matrix T, the attractors of system (D) will be either fixed points or limit-cycles of length at most two (i.e., an iteration loop of length two might occur when using simultaneous updates if there is no other assumption on the T matrix than symmetry). When an input-output nonlinearity is strictly increasing or in the case of quantization nonlinearities, it is shown in [8] that global stability is guaranteed with T having the same assumptions as made by Hopfield [1]-[3] with system (S) updating asynchronously. In [9], a discrete-time neural network $x(k+1)=Tx(k)+I$ is studied with x(k), k=0,1,2,..., confined to a hypercube. By using the energy function $E(x)=-x^T(T-I_d)x-I^Tx$, where I_d is an identity matrix, global stability is guaranteed for system $x(k+1)=Tx(k)+I$, if T is symmetric.

We are now in a position to establish conditions for the global stability of system (S) operating in a synchronous mode.

The neural network with quantization nonlinearity is described by

$$x(k+1)=Tv(x(k))+I \tag{S)'}$$

where $v_i(k)=s_i(x_i(k))$, and $s_i(x_i)=L_r$ when $\tau_r<x_i\leq\tau_{r+1}$, $L_r\in\{L_m,...,L_M\}$, $\tau_r=r-0.5$ (see Fig. 2).

To establish global stability for sytem (S)', we utilize an energy function and show that this energy function is nonincreasing when the state of the network changes.

Define

$$E(k)=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}T_{ij}v_i(k)v_j(k)-\sum_{i=1}^{n}v_i(k)I_i+\sum_{i=1}^{n}c_1G(v_i(k))+\sum_{i=1}^{n}c_2v_i(k),$$

where c_1 and c_2 are constants, and $G(\cdot)$ is a continuous piecewise linear function satisfying equation,

$$c_1\frac{G(L_k)-G(L_{k-1})}{L_k-L_{k-1}}+c_2=\tau_k,\quad k\in\{L_m,...,L_M\}.$$

Theorem 1.3. For the class of discrete-time neural networks described by system(S)', if T is symmetric and positive semidefinite, then system (S)' is globally stable.

1.5. Systems with piecewise linear saturation nonlinearities.

We study a class of systems whose neuron input-output relations are described by piecewise linear saturation functions. An example of such a function is depicted in Fig. 3.

Remark 1.3. 1) The asymptotic stability analysis given in sections 1.1, 1.2, 1.3 is applicable to system (S) with threshold functions of the type shown in Fig.3.

2) $c_{i2}=0$ in a neighborhood of $v_i\in B^n$, $i=1,...,n$.

3) If each "plateau" of a staircase nonlinearity is replaced by a straight line having slope $\varepsilon>0$, where ε is arbitrary small and fixed, then the global stability of (S) is preserved, by Theorem 1.2.

As a consequence of the previous results (given in sections 1.1, 1.2, 1.3), we have the following theorem.

Theorem 1.4. For any $x^\alpha\in B^n$, if

$$s(Tx^\alpha+I)=x^\alpha, \tag{1.5}$$

where $s(x)=(s_1(x_1),...,s_n(x_n))$, $s_i(x_i)$ is of the type shown in Fig. 3, for $i=1,...,n$, then x^α is an asymptotically stable equilibrium point of system (S).

Remark 1.4. Theorem 1.4 provides a means of determining all the possible asymptotically stable equilibrium points in B^n, including the spurious ones.

II. SYNTHESIS

In this section, we adapt some of the results given in [11] to develop a synthesis procedure for system (S). In order to implement system (S) as an effective associative memory, three conditions have to be satisfied:

1) each desired pattern is an equilibrium point of system (S);
2) each desired equilibrium point is asymptotically stable; and
3) the resulting system (S) is globally stable.

Each of the above conditions imposes constraints on system (S). These are considered next.

2.1.1. equilibrium point constraints.

Given is a set of library vectors

$$V=\{v^1,...,v^r\},\quad v^r\in R^n.$$

In order for each $v^i\in V$ to be an equilibrium point of (S), (C1) must be satisfied,

$$x^i=Tv^i+I,\qquad i=1,...,r \tag{C1}$$

where $v^i=(v_1^i,...,v_n^i)$, $x^i=(x_1^i,...,x_n^i)$ and $v_j^i=s_i(x_j^i)$. Equation (C1) is equivalent to (1.1).

To compactly present the design algorithm, we define two matrices V and H as

$$V=[v^1,...,v^r] \tag{2.1}$$

and $$H=[x^1,...,x^r] \tag{2.2}$$

where v^i and x^i are the corresponding i-th output and input library vectors, respectively. In terms of the matrices defined in (2.1) and (2.2), (C1) can be expressed as

$$H=TV+II$$

where $II=[I,...,I]$ is an $n \times r$ matrix with vector I as each of its columns.

The above equation is equivalent to

$$0=TV-H+II.$$

Our goal is to determine T and I so that the linear constraint (C1) is satisfied, when V and H are given. We let

$$R=[V^T,Q]$$

and $W_j=[T_{j1},T_{j2},...,T_{jn},I_j]$

where H_j is the i-th row of H, and $Q \in R^r$ is a column vector containing all ones. Solving (C1) is equivalent to solving

$$H_j^T=RW_j^T \quad \text{for } j=1,...,n. \tag{C1)'$$

When v^i are linearly independent, the solution of (C1)' is

$$W_j^T=PH_j^T=R^T(RR^T)^{-1}H_j^T \tag{2.3}$$

As pointed out in [11], the resulting T matrix is not always symmetric. Next, we will impose the symmetry condition for T, and correspondingly, the global stability condition for (S).

2.1.2. Asymptotic stability constraints.

Constraint (C1) allows us to design a network with each element of V as an equilibrium point. However, this constraint alone does not in general guarantee that each element of V will be asymptotically stable.

Theorem 1.1 in the previous section presents a criterion which can be used to guarantee asymptotic stability. A more easily testable condition equivalent to (A-3) can be formulated in terms of M-matrices:

If there exist constants $\lambda_j>0$, $j=1,...,n$, such that

$$\Sigma_{j=1}^{n}\lambda_j D_{ij}>0, \qquad \text{for } i=1,...,n \tag{2.4}$$

then the equilibrium point p=0 of system (Σ) is asymptotically stable.

If we pick $\lambda_j=1$, for $j=1,...,n$, (2.4) assumes the form

$$1-\Sigma_{j=1}^{n}|T_{ij}|c_{j2}>0, \quad \text{for } j=1,...n. \tag{C2}$$

Remark 2.1. If the input-output characteristic $s(\cdot)$ in the neighborhood of p=0 is very "flat", i.e., the c_{j2}'s are very small, then (C2) is satisfied. This is in particular true when the piecewise linear saturation characteristic function is used. In this case $c_{j2}=0$ in a neighborhood of p=0.

2.1.3. Global stability constraints.

From Theorems 1.2 and 1.3, we know that when the interconnection matrix T of system (S) is symmetric and positive semidefinte, the global stability of system (S) is guaranteed when appropriate input-output nonlinearities are used. When T is a positive semidefinte matrix, we denote it as

$$T \geq 0. \tag{C3}$$

2.2. The synthesis procedure.

This section presents a method for designing a network which satisfies (C1) (C2) and (C3). The solution which satisfies (C1) is given in (2.3). However, the T_{ij}'s resulting from equation (2.3) may not necessarily be symmetric. In the following we provide a condition under which the symmetry of the interconnection matrix T is guaranteed. It is stated as the following proposition.

Proposition 2.1. For the given sets of library vectors V and H, if v^i and x^i satisfy

$$v^i=s(x^i) \tag{2.5.1}$$

and

$$v^i=x^i, \tag{2.5.2}$$

for $i=1,...,r$ (e.g., at point p* in Fig. 4, $v^i=x^i$), then equation (2.3) will yield a symmetric interconnection matrix, i.e., $T_{ij}=T_{ji}$.

Remark 2.2. For the input-output characteristics shown in Fig.1, Fig.2 and Fig.3, the condition $v^i=x^i$, $i=1,\dots,r$, is satisfied without requiring additional assumptions in the design of system (S).

The constraint (C2) is satisfied by picking an input-output characteristic function which is "sufficiently flat" in a neighborhood of the equilibrium point $p=0$.

Next, we make further assumptions on system (S) in order to make the interconnection matrix T positive semidefinite. Under these conditions, the global stability of system (S) is guaranteed.

Theorem 2.1. For the given sets of input and output library vectors V and H, if (2.5.1) and (2.5.2) are satisfied, and if the bias term I of system (S) is zero, the T matrix given by

$$T=VV^{+} \tag{2.6}$$

where V^{+} is the pseudo-inverse of V, is positive semidefinite.

The conditions (C1), (C2), and (C3) are satisfied by the following synthesis procedure:

Step1. Determination of a threshold function.

A neuron input-output nonlinearity has to intersect the line $s(x)=x$ on the "plateaus" of a threshold function (see, e.g., Fig.4). In doing so, the gain of a staircase nonlinearity (Fig. 3) has to be greater than 1.

Step 2. Determination of input library vectors.

The input library vectors are uniquely determined by equations (2.5.1) and (2.5.2), i.e., $x^i=v^i$, $i=1,\dots,r$, for a given set of output library vectors $\{v^i\}$.

Step 3. Determination of T and I matrices.

Substituting V and X into equation (2.3), the T and I matrices are determined. Alternatively, T may be calculated from equation (2.6) with $I=0$. In the latter case, the designed system (S) will be globally stable for the threshold nonlinearities shown in Fig. 1 and Fig. 2.

III. EXAMPLES AND APPLICATIONS

In the present section, we present two examples using the synthesis procedure given in Section 2. The purpose of the first example is to demonstrate the details of the design procedure. In the second example we demonstrate the applicability of the present synthesis procedure to image processing. In this example, a neural network has been designed to classify an image composed of 256×256 pixels, each pixel being capable of assuming integer values ranging from 0 to 15. A sixteen stage staircase input-output nonlinearity is used in this example.

Example 3.1.

We first design a very simple two neuron system to demonstrate that the desired equilibrium points have been stored as asymptotically stable equilibrium points and to use Theorem 1.4 to determine spurious asymptotically stable equilibrium points in B^n.

For a two neuron system, the given desired library vectors are,

$$v^1=\begin{bmatrix}0\\1\end{bmatrix}, \text{ and } v^2=\begin{bmatrix}2\\2\end{bmatrix}.$$

The input-output nonlinearity used is given in Fig.3 with the five output values 0, ±1, ±2.

We choose, for the sake of simplicity,

$$x^1=\begin{bmatrix}0\\1\end{bmatrix}, \text{ and } x^2=\begin{bmatrix}2\\2\end{bmatrix}.$$

Substituting V and X into equation (2.3), we obtain

$$T=\begin{bmatrix}0.8889 & 0.2222\\0.2222 & 0.5556\end{bmatrix}, \text{ and } I=\begin{bmatrix}-0.2222\\0.4444\end{bmatrix}.$$

By Theorem 1.4, the spurious asymptotically stable equlibrium points in B^n are determined as

$$\alpha^1=\begin{bmatrix}1\\2\end{bmatrix}, \alpha^2=\begin{bmatrix}-1\\1\end{bmatrix}, \alpha^3=\begin{bmatrix}1\\1\end{bmatrix}, \alpha^4=\begin{bmatrix}-2\\0\end{bmatrix}, \alpha^5=\begin{bmatrix}-1\\0\end{bmatrix}.$$

Example 3.2. (Application to image processing.)

We demonstrate the applicability of system (S) and the present synthesis procedure by means of a specific grey level image processing example. In doing so, a 4096 neuron network is designed. We wish to store three 256×256 pixel images. Because of computer memory limitations, we partition each image into sixteen parts, in order to store one complete image. For each image, the sixteen subdivided parts of the image are stored as equilibrium points.

Thus a total of 48 equilibrium points are desired for system (S), with 4096 neurons. The three images are quantized from the original 8bit/pixel images into sixteen grey level images. Thus each neuron in system (S) has sixteen possible outputs ranging from 0 to 15. In accordance with the notation given in the first section, we have n=4096, r=48, L=16, and b=4. The purpose of the synthesized system is to recover the key patterns from noisy input images. Gaussian noise is used in our simulations. Noisy input patterns are generated by adding zero mean Gaussian noise with a given specific standard deviation to the key images. In particular, the image of Fig.6, which was stored by 16 equilibria in the neural network is corrupted by zero mean Gaussian noise with a standard deviation of 1.5 to yield the image given in Fig. 5. Using this noisy image as initial condition, the neural network converged to the original image given in Fig. 6 in 15 iterations.

IV. CONCLUSIONS

In this paper, we have proposed a neural network model endowed with multi-level threshold functions as an effective means of realizing associative memories. We have conducted a qualitative analysis of these networks and we have devised a synthesis procedure for this class of neural networks. We applied the present results to a specific grey level image processing example. Simulation results (not included in the present paper) suggest that the smaller the slope of the staircase nonlinearity for each neuron (see, e.g., Fig. 3), the larger the domain of attraction of each desired equilibrium point.

REFERENCES

[1]J. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties like Those of Two-state Neurons", Proc. Nat'l. Acad. Sci., U.S.A., Vol. 81, May 1984, pp. 3088-3092.

[2]J. J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems", Biological Cybernectics, Vol. 52, 1985, pp. 141-152.

[3]D. W. Tank and J.J. Hopfield, " Simple 'Neural' Optimization Networks: an A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", IEEE Trans. On Circuits and Systems, Vol. CAS-33, No.5, May 1986, pp. 533-541.

[4]L. Personnaz, I. Guyon and G. Dreyfus, "Information Storage and Retrieval in Spin-glass like Neural Networks", J. Physique Letters, Vol. 46, April 1985, pp. L359-L365.

[5]L. Personnaz, I. Guyon and G. Dreyfus, "Collective Computational Properties of Neural Networks: New Learning Mechanisms", Physical Review A, The American Physical Society, Vol. 34, No. 5, Nov. 1986, pp. 4217-4228.

[6]W. Banzhaf, "Towards Continuous Models of Memory", Proceedings of the IEEE First International Conference on Neural Nets, San Diego, Vol. 2, June 1987, pp.223-230.

[7]A. Guez, V. protopopsecu and J. Barhen, "On the Stability, Storage Capacity, and Design of Nonlinear Continous Neural Networks", IEEE Trans. on System, Man and Cybernetics, Vol. 18, No.1, Jan./Feb., 1988, pp.80-86.

[8]M. Fleisher, "The Hopfield Model with Multi-Level Neurons", American Institude of Physics, 1987, pp. 278-289.

[9]A. N. Michel, J. Si, and G. Yen,"Analysis and Synthesis of a Class of Discrete-Time Neural Networks Described on Hypercubes", going to appear in IEEE Trans. on Neural Networks.

[10]J. Li, A. N. Michel and W. Porod, "Analysis and Synthesis of a Class of Neural Networks : Linear Systems Operating on a closed hypercube", IEEE Trans. on Circuits and Systems, Vol.36, No.11, Nov. 1989, pp. 1405-1422.

[11]A. N. Michel, J. A. Farrel and H. F. Sun, "Analysis and Synthesis Techniques for Discrete Time Neural Network Models", Proceedings of the 1989 IEEE Conference on Decision and Control, Tampa, Florida, Dec. 1989.

[12]A. N. Michel, J. A. Farrel, "Associative Memory via Artificial Neural Network Models", IEEE Control Systems, Vol. 10, No.3, April 1990, pp. 6-17.

[13]J. Bruck, J. Goodman, "A Generalized Convergence Theorem for Neural Networks", IEEE Trans. on Information Theory, Vol. 34, No.5, Sept. 1988, pp.1089-1092.

[14]E. Goles, F. Fogelman and D. Pellegrin, "Decreasing Energy Functions as a Tool for Studying Threshold Networks", Discrete applied Mathematics, 12 (1985), pp.261-277.

[15]A. N. Michel, and R.K. Miller, Qualitative Analysis of Large Scale Dynamical System, Academic Press, New York, 1977.

[16]A.N.Michel, "On the Status of Stability of Interconnected Systems", IEEE Trans. on Automatic Control, Vol. AC-28, No.6, June 1983, pp. 639-653.
[17]M.A.Pai, Power System Stability, Amsterdam, The Netherlands: North Holland, 1981.
[18]R.K.Miller, A. N. Michel, Ordinary Differential Equations, Academic Press, New York, 1972.
[19]A. Albert, Regression and the Moore-Penrose Pseudo-Inverse, Academic Press, 1972.

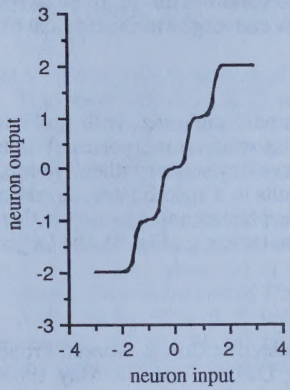


Fig. 1 nulti-level sigmoidal function

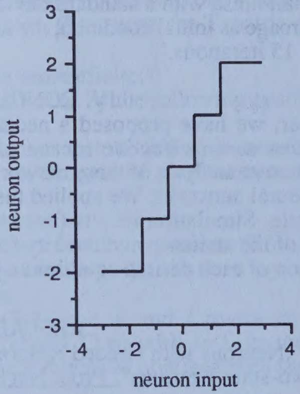


Fig. 2 quantization nonlinearity

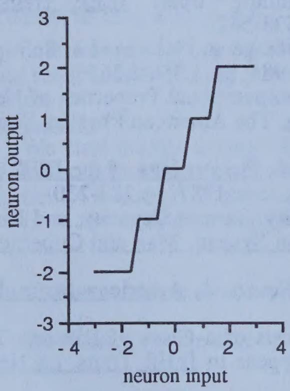


Fig. 3 staircase nonlinearity

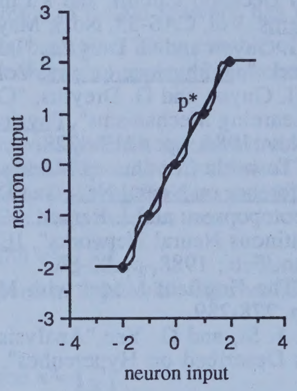


Fig. 4 location of an asymptotically stable equilibrium

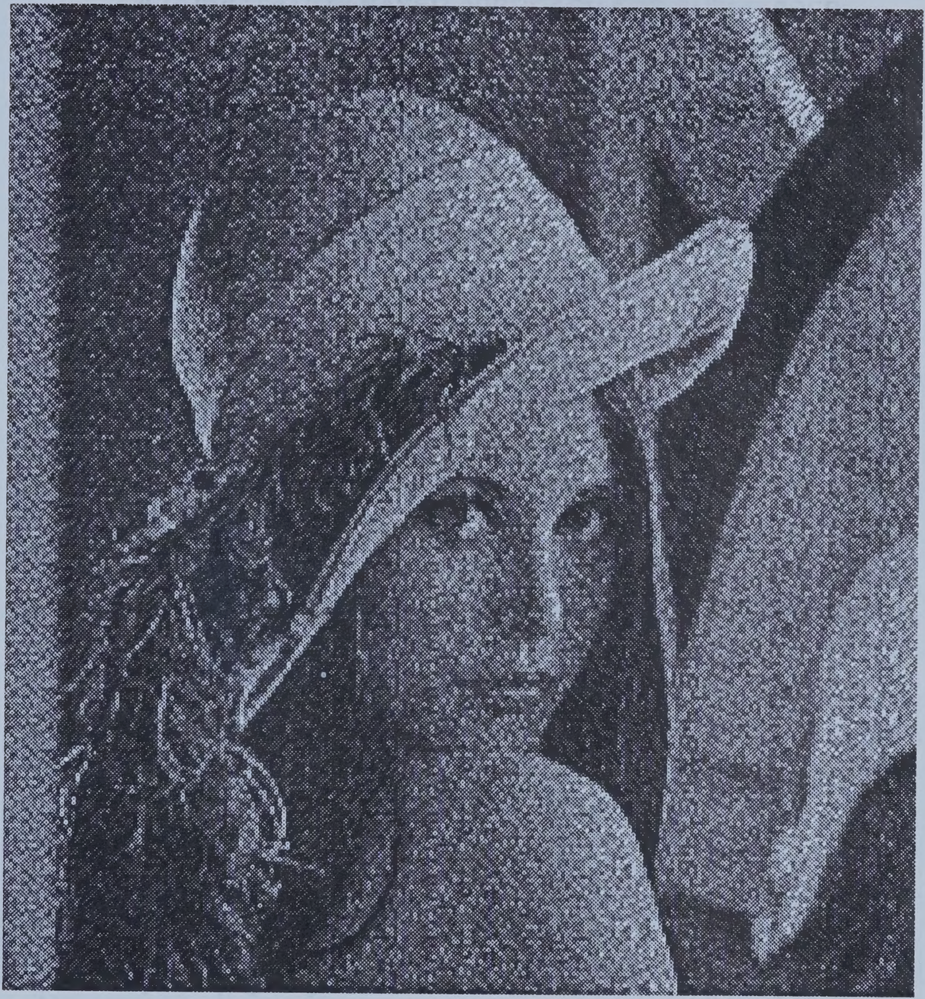


Fig. 5 noisy image as initial condition for the recovery of the oringinal image

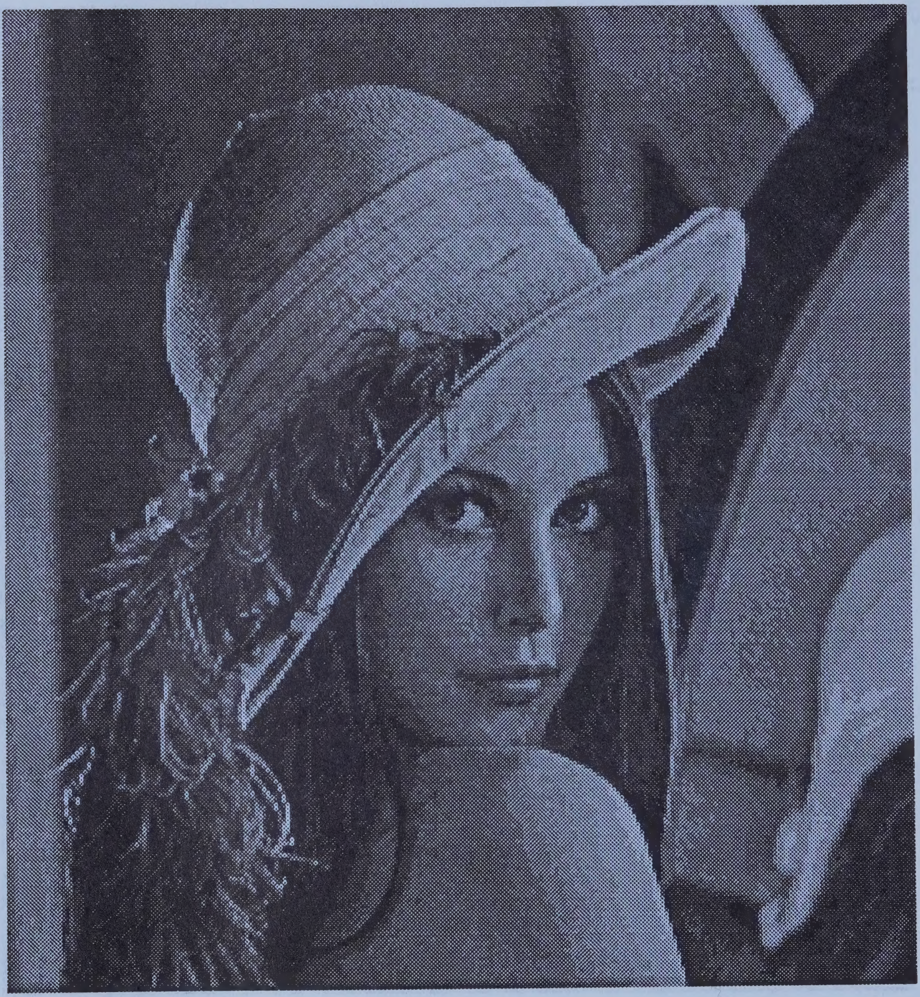


Fig. 6 original image stored in system (S)

A CORRELATION-BASED APPROACH FOR SUPERVISED NEURAL-NET CLASSIFIER

R. Y. Li and M. Xu
Department. of Electrical Engineering.
University of Nebraska
Lincoln, NE 68588

Abstract

In general, the training of a conventional backpropagation (CBP) neural network is very slow because the process begins with random initial weights and an arbitrarily assigned hidden layer structure. An algorithm called Thresholded Correlation Learning Vector Analysis (TCLVA) can sharply cut down the learning time by using a correlation-based approach to estimate a good set of initial weights, biases, and network size. The simulation results show that this new approach indeed yields superior performances.

1.0 Introduction

In the training of a conventional backpropagation (CBP) net, the learning process begins with random initial weights, biases, and an arbitrarily assigned hidden layer structure. In general, the initial weights and biases are far away from a set of weights and biases which can produce an acceptable classifation result. Therefore, too many learning iterations are required[1,2]. To speed up the learning of a neural network, it is necessary that a good set of initial weights and biases is assigned to the network, and a proper hidden layer size is determined. An algorithm called Thresholded Correlation Learning Vector Analysis (TCLVA) is presented for that purpose. The simulation results show that this neural-net classifier indeed yields much better performances .

2.0 Thresholded Correlation Learning Vector Analysis(TCLVA)

2.1 Principle of TCLVA

The neural network using the correlation approach has the following assumptions:

(1) The weights at the input side of a hidden unit form a representative vector of the patterns that this particular hidden unit can recognize. This hidden unit can reject patterns from all other classes, using the correlation value between its weight vector and an input pattern vector as the decision factor.

(2) There are enough hidden units to cover all classes. We also expect that the network can classify correctly all the learning patterns on the output layer of the network by modifying weights and biases between the hidden and output layers.

To estimate the similarity between patterns, it is necessary to normalize all the pattern vectors. Then, the correlation between two normalized vectors I and Wj is:

$$R_j = \sum^{K} I_K * W_{jK} \qquad (1)$$

and $-1 < R_j < +1$, where K is the vector dimension and R_j is the correlation value between two normalized vectors at a hidden unit j. The more similar the two normalized vectors are, the closer R_j is to 1. If they are extremely dissimilar, then R_j can approach -1.

In order to let a hidden unit, which belongs to class h, to reject all other classes , there should be a proper threshold t_d which is chosen to be the maximum value of correlations between all other classes' patterns and the weight vector W_j. Thus

$$t_d = \max [R_j] \qquad \text{for} \qquad I \notin C_h \qquad (2)$$

The output of a sigmoid function x (See Figure 1) of a hidden unit is designed to be greater than or equal to Q1 when a learning pattern of the same class is presented. It is to be less than or equal to Q2 when a pattern of different class is presented. We chose Q1 to be 0.9 and Q2 to be 0.1. Another threshold t_u is calculated as:

$$t_u = m * (1 - t_d) + t_d \qquad (3)$$

where m is a positive fraction number to make an interval $[t_d, t_u]$. For example, if m = 0.05, we would have

$$-1 < t_d < t_u < +1$$

Then the bias term of the hidden unit is:

$$B_j = 0.5 * (t_d + t_u) \qquad (4)$$

The purpose is to move the midpoint (0.5) of the sigmoid function from origin to the midpoint of the interval. There should be a scale factor C_j for the hidden unit so that the absolute value of the net input for the sigmoid function is greater than or equal to a positive constant Q3. The purpose is to create a desired output, i.e. $x > Q1$ or $x < Q2$ for a given learning pattern.

$$C_j = 2 * Q3 / (t_u - t_d) \qquad (5)$$

Q3 is chosen to be 2.1973 if Q1 = 0.9 and Q2 = 0.1. Finally, the weights and biases for the hidden unit are assigned as

$$W_{jk} = C_j * I_k$$

$$B_j = C_j * B_j \qquad (6)$$

Each hidden unit has different t_d, t_u, B_j and C_j. A smaller t_u can cover more input space for a given class, but it can also lead to a greater error. This case can be controlled by adjusting the value of m. In an extreme case where $t_u = 1$, all the learning patterns will generate their own hidden units.

We call this method Thresholded Correlation Learning Vector Analysis (TCLVA) because it is based on the correlation among learning vectors, and a set of thresholds. It should be noted that if a given pattern can be correctly classified by the net using the TCLVA algorithm, there is at least one hidden unit at which the output of the sigmoid function x is greater than or equal to Q1.

2.2. TCLVA Algorithm

The TCLVA algorithm can provide a 3-layer net with a set of optimal weights and biases between the input and hidden layers automatically, and can choose an adequate number of hidden units naturally from the training patterns. The complete procedure is outlined below:

(1) Normalize all the input patterns. Mark all the learning patterns by using 0 flags.

(2) Starting from an arbitrary class, choose randomly from this class a learning pattern which has a 0 flag, to set up the j-th new hidden unit. For the first hidden unit, j is equal to 1. The class flag for the hidden unit is set up at the same time, and it should be consistent with the class of the chosen learning pattern. Assign elements of the learning pattern as the elements of the weight vector for that hidden unit. If all patterns for a given class have 1 flags, then choose patterns from the next class group.

(3) Scan R_j for all patterns of different classes by Eq. (1). Calculate t_d by Eq.(2), and t_u by Eq.(3).

(4) Scan the correlation values between the weight vector and all learning patterns with a zero flag in the same class. If $R_j > t_u$, mark pattern I by a 1 flag. The pattern with a 1 flag has been recognized and will not be considered further as a possible weight vector for the new hidden unit.

(5) Search a new learning pattern with a zero flag. If there is any one left in the set of learning patterns, then j = j+1, and go back to step (2). Otherwise go to step (6).

(6) Now all representative vectors for the set of learning patterns have been utilized. The TCLVA algorithm further adjusts the weights and biases for all hidden units by Eq. (4), (5) and (6).

(7) Then assign random weights and biases between the hidden and output layers. Perform standard backpropagation training until the final network output converges to a specified error tolerance for all the training patterns.

3.0 Simulation and Results

We apply CBP and TCLVA neural network models to multispectral image data from a 10-band airborne spectrometer dataset . In our experiments, the learning rate constant η is chosen as 0.9, and the momentum constant α is chosen as 0.7.

In the experiment , two sets of data come from the CALMIT laboratory of the University of Nebraska[3]. These data with 10 spectral bands have been divided into 5 known classes. They are water, vegetation, non-vegetated farmland, wet soil, and wetland vegetation. Figure 2 shows a typical multispectral image with overstriking characters corresponding to different gray levels[5]. The training set has 923 patterns, and the testing set has 815 patterns. Table 1 lists some data patterns for the 5 classes. Table 2 lists the normalized mean vectors for the 5 classes.

We tested a total of 4 networks. The first three networks use CBP configurations with different numbers of hidden units. The last network simulates TCLVA configuration. Using random initial weights and biases for the

CBP nets with different hidden layer sizes, we compared the difference in terms of their learning speeds. For the TCLVA net, we checked whether it can create optimal weights and improve learning speed. As a comparison against the standard method[4], we used the Bayesian classifier on the same data set.

From Table 3, the TCLVA classifier has the fastest rate of convergence. The main reason is that it only has to train the weights and biases between the hidden and output layers. Also from Table 3, the TCLVA net only requires 23 iterations, compared with 3438 iterations for the CBP net #3. The error curves for these two cases are shown in Figure 3 and Figure 4, respectively. In addition, the 23 iterations are performed only between the hidden and output layers, requiring even less time per iteration in comparison with the CBP model. Of course, the training effort for determining weights between input and hidden layers should not be ignored. It is also found that the recognition rates of all the neural-net classifiers are comparable to that of the Bayesian classifier.

4.0 CONCLUSIONS

In our experiments, the CBP network was able to classify multi-dimensional patterns, such as the multispectral data. In general, the CBP model requires a tremendous number of learning iterations to develop the capability to classify patterns because random initial weights are used. The TCLVA algorithm can sharply cut down the learning time by using correlation-based approach to estimate initial weights, biases, and network size. The classification accuracy of a TCLVA network is comparable to that of a Bayes classifier. In general, the neural-net classifier does not need an assumption about the statistical distribution of the data patterns and can continue to learn new patterns based on its current state. In this regard, it is more versatile than the Bayesian approach.

5.0 REFERENCES

[1] Richard P. Lippmann: "AN INTRODUCTION TO COMPUTING WITH NEURAL NETS", IEEE ASSP MAGAZINE April 1987, PP.4-21.

[2] D. E. Rumelhart et al., PARALLEL DISTRIBUTED PROCESSING , The MIT Press (1986), vol. 1.

[3] Donald C. Rundquist et al., "BAND-MOMENT ANALYSIS OF IMAGE SPECTROMETER DATA", Photogrammetric Engineering and Remote Sensing.vol. 55, No.2, 1989, pp.203-208.

[4] Yoh Han Pao, ADAPTIVE PATTERN RECOGNITION AND NEURAL NETWORKS, Addison-Wesley, pp.26-50 1989.

[5] Rafael C. Gonzales and Paul Wintz, DIGITAL IMAGE PROCESSING, Second Edition, pp.421-423 1987.

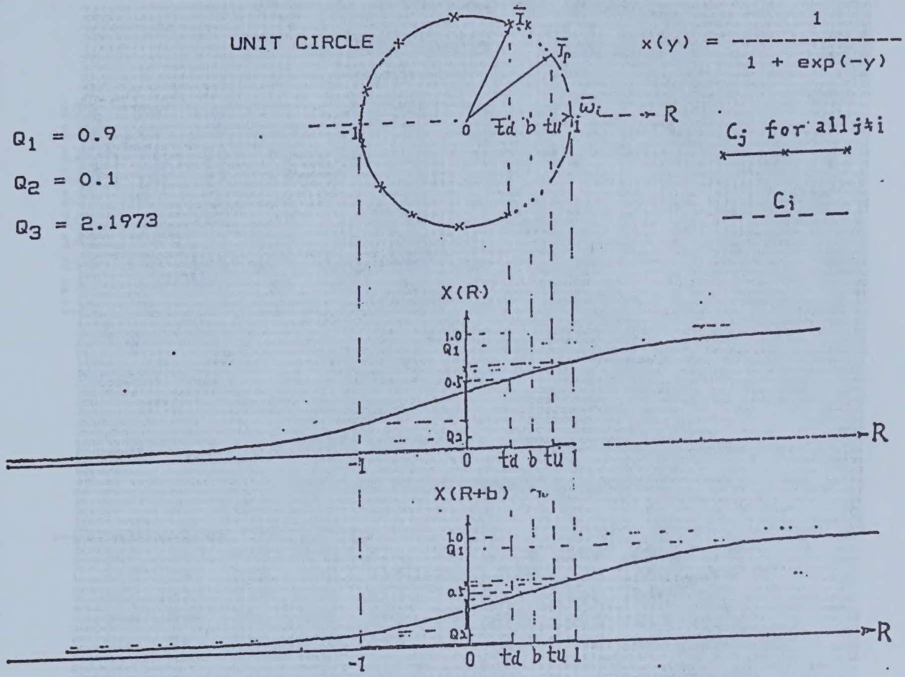

UNIT CIRCLE
$\bar{I}_k$
$\bar{I}_p$
$\bar{\omega}_i$
R
0
td
b
tu
1
-1
$x(y) = \frac{1}{1 + \exp(-y)}$
$Q_1 = 0.9$
$Q_2 = 0.1$
$Q_3 = 2.1973$
C_j for all j≠i
C_i
X(R)
1.0
Q_1
0.5
Q_2
X(R+b)

FIGURE 1 FORMATION OF A WEIGHT VECTOR AND A BIAS FOR A HIDDEN UNIT

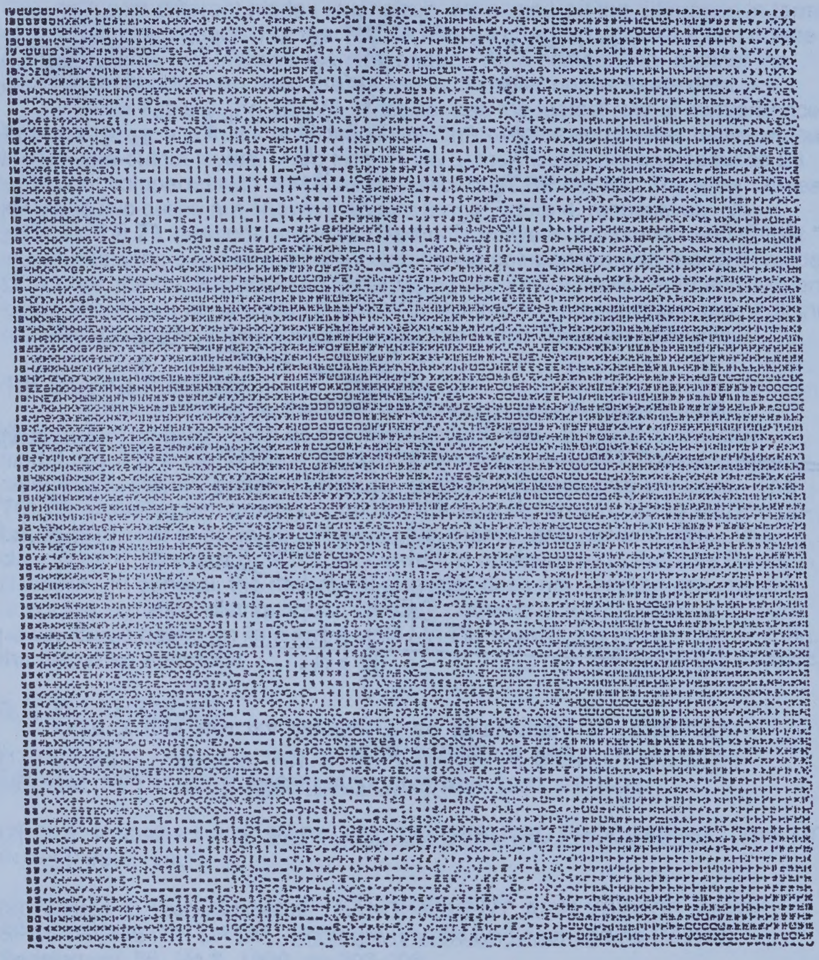


FIGURE 2   A MULTISPECTRAL IMAGE

TABLE 1 TYPICAL PATTERNS

FOR THE MULTISPECTRAL IMAGE DATA

										CLASS
132	84	108	71	66	74	85	45	25	23	1
139	84	109	72	66	75	86	45	25	22	1
121	74	90	55	59	70	80	88	68	97	2
117	72	83	45	51	64	69	90	71	100	2
123	65	74	32	35	46	35	20	13	19	3
103	65	74	32	36	46	35	20	12	16	3
142	96	131	100	89	95	125	76	56	83	4
155	105	145	113	100	107	141	84	61	89	4
123	80	100	65	60	68	80	48	38	57	5
131	77	94	55	52	62	66	40	31	51	5

TABLE 2 NORMALIZED MEAN VECTORS FOR THE 5 CLASSES

OF THE MULTISPECTRAL DATA

										CLASS
.535	.340	.435	.290	.270	.300	.335	.165	.090	.080	1
.490	.295	.355	.200	.215	.265	.290	.325	.250	.355	2
.655	.375	.435	.200	.215	.275	.220	.120	.085	.120	3
.455	.305	.410	.300	.275	.295	.370	.225	.165	.245	4
.550	.345	.420	.260	.240	.280	.310	.190	.145	.230	5

TABLE 3 CLASSIFICATION RESULTS OF MULTISPECTRAL IMAGE DATA FOR CBP NET, TCLVA NET, AND BAYESIAN CLASSIFIER

ALGORITHM	CBP			TCLVA	BAYES
NET NO.	1	2	3	7	-
NO. OF UNITS INPUT LAYER HID. LAYER OUT. LAYER	18 10 5 3	20 10 7 3	22 10 9 3	22 10 9 3	-
SELECTION OF INITIAL WEIGHTS	RANDOM	RANDOM	RANDOM	AUTOMATICALLY SEARCH	-
ITERATION NO. ON LEARNING SET OF PATTERNS (923)	5771	4519	3438	23	
ERROR	0	0	0	0	0
ERROR ON TESTING PATTERNS (815)	2	3	2	1	0
REC. RATE %	99.7546	99.6319	99.7546	99.8773	100.000

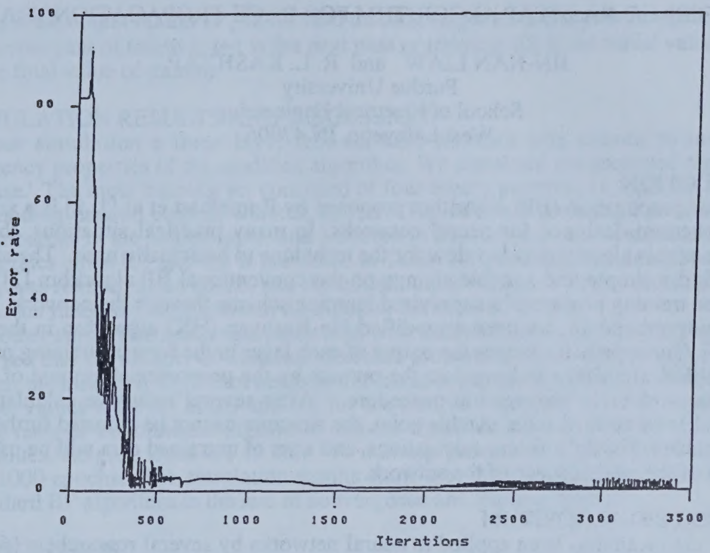


FIGURE 3 ERROR-ITERATION CURVE FOR CBP NET #3

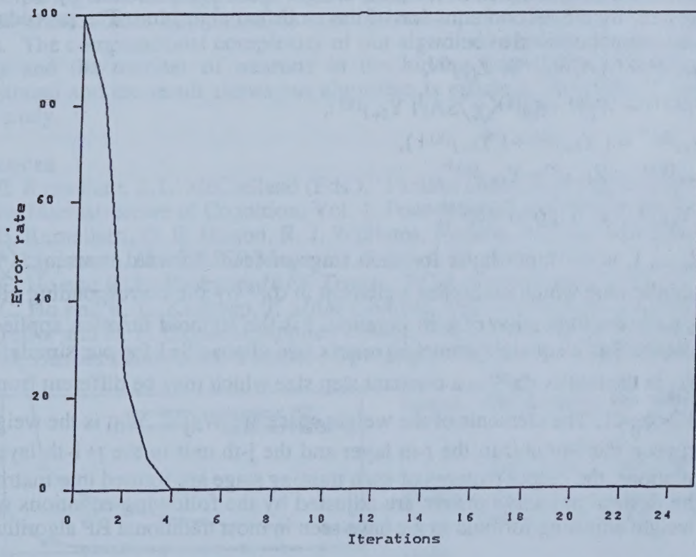


FIGURE 4 ERROR-ITERATION CURVE FOR TCLVA NET #7

A MODIFIED HO-KASHYAP ALGORITHM FOR BACK-PROPAGATION TRAINING

JIN-NAN LIAW and R. L. KASHYAP
Purdue University
School of Electrical Engineering
West Lafayette, IN 47906

I. INTRODUCTION

The back propagation (BP) algorithm proposed by Rumelhart et al [1, 2] is a supervised learning procedure designed for neural networks. In many practical situations, the rate of convergence of this algorithm is too slow for the technique to be actually used. The aim of our work is to find a simple and suitable change on the conventional BP algorithm [3] that will accelerate the training process of a supervised learning scheme through the network. To speed up the learning procedure, we used a modified Ho-Kashyap (HK) algorithm in the training scheme [4,5]. Our approach changes the output of each layer in the forward training process by the modified HK algorithm and regulate the outputs by the proportional amount of errors in each layer through error propagation procedure. After several recursive calculations, the network reaches an optimal state. At this point, the structure cannot be adjusted further. Then, the weight space is fixed for future calculations, and a set of untrained data will be used to test the accuracy of the weight space of the network.

II. THE MODIFIED ALGORITHM

The HK algorithm has been applied to neural networks by several researchers [6,7]. Here we applied a modified HK algorithm to a multilayer feedforward neural network. Consider a feedforward neural network with one input layer, one output layer and L - 1 hidden layers. Let N_l be the number of neurons in each layer, $l=1$ is the input layer and $l=L+1$ the output layer. Suppose we have M sets of input data which will be trained through the network to adjust the weight space. According to the matrix form of the HK algorithm, the matrix A consists of the M sets of input data with dimension $M \times N_l$; the weight space is represented by $W_l^{(k)}$ which is iteratively adjusted by the second equation of the modified algorithm. The procedure of the modified algorithm can be stated as below:

1. $Y_{l+1}^{(k)} = f\{A_l W_l^{(k)}\} - Z_{l+1}^{(k)}$,
2. $W_l^{(k+1)} = W_l^{(k)} + \Phi_l^{(k)} \langle\chi\rangle S A_l^t | Y_{l+1}^{(k)} |$,
3. $Y_{l+1}^{(k)+} = (Y_{l+1}^{(k)} + | Y_{l+1}^{(k)} |)$,
4. $Z_{l+1}^{(k+1)} = Z_{l+1}^{(k)} + Y_{l+1}^{(k)+}$,
5. If $Y_{l+1}^{(k)+} \neq 0$, go to step 1,

where $l=1, 2, \ldots, L$ is the input layer for each stage of feed- forward training; $\langle\chi\rangle$ is the Hadamard multiplication which multiplies a element in $\Phi_l^{(k)}$by the corresponding element in $S A_l^t | Y_{l+1}^{(k)} |$; (k) is the indication of k-th iteration. f is the sigmoid function applies to each element of $A_l W_l^{(k)}$. S is a square symmetric matrix, we choose S=I for our simulation later. Each element ϕ_{ij} in the matrix $\Phi_l^{(k)}$ is a constant step size which may be different from neuron to neuron, and $0<\phi_{ij}<1$. The elements of the weight space are W_{lij}'s. W_{lij} is the weight of the connection between the i-th unit in the l-th layer and the j-th unit in the l+1-th layer. After recursive calculations, the output patterns of each training stage are formed into matrix $Z_{l+1}^{(k)}$. Elements of the desired objective matrix are adjusted by the following equations which are similar to the weight adjusting formula as we have seen in most traditional BP algorithm.

$$Z_l^{(0)}(t+1) = Z_l^{(*)}(t) + \eta E_l \langle\chi\rangle O_l^{(*)} + \alpha \Delta Z_l^{(*)}(t) \tag{1}$$

$$\Delta Z_l^{(*)}(t) = Z_l^{(*)}(t) - Z_l^{(*)}(t-1) \tag{2}$$

where o_l is the output matrix of l-th forward training stage; t-1 is the previous pass of training; t is the current pass of training; t+1 is the next pass of training; (0) is the initial value of training; (*) is the final value of training

III. SIMULATION RESULTS AND DISCUSSION

In our simulation a three layer feed-forward network was chosen to investigate the convergency properties of the modified algorithm. We simulated the proposed algorithm on an XOR case. The input training set consisted of four binary patterns: (1,1),(0,0),(1,0),(0,1); the corresponding desired outputs were: 0, 0, 1, 1. The network includes 2 neurons in the first layer, 2 neurons in the hidden layer, and 1 neuron in the output layer. A simulation result with learning rates equal 0.7 and 0.3 is presented in Table 1. A training iteration through the entire network was referred as an epoch, and the initial output values of hidden units for each pattern were chosen randomly with uniform distribution between 0.1 and 0.

Another simulation result was performed with different learning rates. In Figure 1, a trend is observed that the number of iterations decrease proportionally as the learning rate increases. However, the modified algorithm needs fewer iterations than the standard one does. The limits for target values were set as 0.9 and 0.1 for the desired values of 1 and 0, respectively. The learning rates of the classical algorithm and the modified algorithm both were assigned the same values, ranging from 0.1 to 1.0. A training process was considered if it converged within 1000 epochs. The simulation results show that this modified algorithm is superior to the standard BP algorithm in the rate of convergence and learning time.

IV. CONCLUSION

We have introduced a new BP algorithm in which a modified Ho-Kashyap procedure was applied to speed up the training process. The learning rate of our new algorithm appears to be faster than that of a conventional BP algorithm. Our network was trained on an output-updating recursive calculation scheme which has reduced the computational burden of a standard training process.

Comparing this new approach with a standard BP algorithm, we found that the new approach in most cases approximated to an optimal solution with fewer iterations in a training process. The computational complexity of our algorithm depends on the size of the training patterns and the number of neurons in the hidden layer. The simulation on XOR is demonstrated and the result shows our algorithm is efficient; therefore, it could be used for further study.

References

1. D. E. Rumelhart, J. L. McClelland (Eds.), "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations", MIT Press 1986.
2. D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Nature*- 323, pp. 533-536, Oct. 1986.
3. R. P. Lippmann, *IEEE ASSP Magazine, 4*: pp. 4-22, April 1987.
4. Y. C. Ho and R. L. Kashyap, *IEEE Trans.* , EC-4, pp. 683-688, 1965.
5. Y. C. Ho and R. L. Kashyap, J. *SIAM Control*, Vol. 4, No. 1,pp. 112-115, 1966.
6. B. Telfer and D. P. Casasent, *Applied Optics*, Vol. 29, No. 8, pp. 1191-1202, Mar. 1990.
7. M. H. Hassoun and A. M. Youssef , *Opt. Eng*., Vol. 28, No. 1, pp.46-54, January 1989.

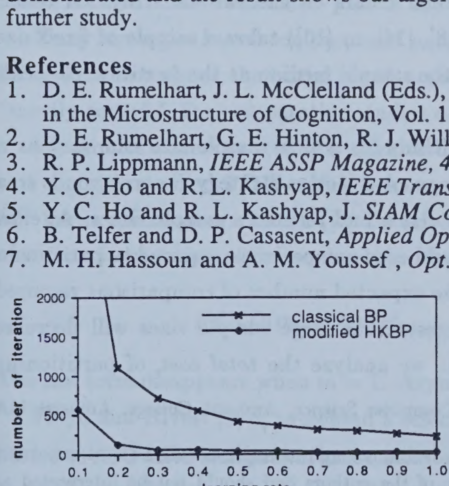


Figure 1: No. of iterations for different learning rates.

Table 1: XOR simulation with different learning rates

	no. of runs	learning rate	epochs			
			max.	min.	mean	var.
modified K-H BP	49	0.3	120	2	49	25
	49	0.7	60	2	25	11
classical BP	49	0.3	513	346	397	33
	49	0.7	221	149	171	14

Optimal Sampling Strategies for Quicksort

C. C. McGeoch* J. D. Tygar [†]
DIMACS Center School of Computer Science
Rutgers University Carnegie-Mellon University

Abstract

A well-known variant of Quicksort samples from the subarray at each recursive stage and uses the sample median as the partition element. Three families of *general sampling strategies*, which allow sample size to vary as a function of subarray size, are considered. The total cost of sorting comparisons plus median-selection comparisons is analyzed. A strategy which takes a sample of size $\theta(\sqrt{n})$ for subarray size n is shown to be optimal over a large class of strategies. The square-root strategy has $O(n^{1.5})$ worst-case cost. The true optimal strategy is found computationally for $n < 10,000$.

1 Introduction

The basic operation of Quicksort partitions an array $X = x_1, x_2, \ldots, x_n$ around a partition element $p = x_i$ so that elements with value less than p are to its left in the array and larger elements are to its right; Quicksort then recurs on the two subarrays on either side of p. A well-known generalization (see [8], [14], or [13]) takes a sample of fixed size (usually 3) at each recursive stage and uses the sample median as the partition element; we call this a *fixed-sample (FS) strategy.*

We introduce a class of Quicksort algorithms for which sample size can vary as a function of subarray size at each recursive stage. A selection strategy S corresponds to a *sampling function* $\sigma(n)$ which takes subarray size n and returns a sample size s. At each recursive stage, the *partitioning cost* is the number of comparisons required to partition a subarray of size n and the *selection cost* is the expected number of comparisons required to find the median s items. Intuition suggests that large sample sizes will decrease partitioning cost but increase selection cost: we analyze the *total cost*, of partitioning

---

*Permanently at: Dept. of Mathematics and Computer Science, Amherst College, Amherst MA 01002.

[†] Supported by National Science Foundation Presidential Young Investigators Grant CCR-88-508087. The views and conclusions in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation or the U. S. Government.

plus selection, summed over all recursive stages. This cost model and the generalization to varying sample sizes are new to the analysis of Quicksort.

We present the following results:

- A *two-tier (TT) strategy* takes a sample of size $s = \sigma(n)$ at the first stage only; at lower levels of recursion the sample is of fixed size t. We derive an exact formula for the optimal sampling function $\sigma(n)$ for any t. For any problem size N the optimal *TT* strategy has lower total cost than any *FS* strategy.
- A *square-root (SQ) strategy* takes a sample of size $\theta(\sqrt{n})$ at all recursive stages. We describe a square-root strategy which has lower total cost than any TT strategy.
- Any SQ strategy requires $O(n^{1.5})$ total comparisons in the worst case. This is the only Quicksort-based algorithm that we know of with subquadratic worst-case cost.
- For given problem size we can compute exactly the optimal strategy; we compare this strategy to standard fixed-sample strategies for $N < 10000$.

The original Quicksort algorithm, proposed by Hoare [5, 6] selected the leftmost subarray element at each recursive stage. Hoare noted that a randomly-selected partition element would ensure expected-case behavior and suggested that larger samples might improve performance. Singleton [16] and Sedgewick [14, 13] considered FS strategies and recommended *median-of-3* Quicksort where sample size t equals 3 at all recursive stages. The analyses were not made precise because selection cost was not an explicit part of the model; rather it was considered part of the overhead of each recursive call.

Many strategies have been proposed to improve performance of Quicksort when something is known about the initial input permutation, for example when the elements are "mostly sorted." For comparative studies of such variations, see [1], [9] and [10].

We require a median-selection algorithm as a subroutine. Hoare [5] presented a selection algorithm called FIND which behaves similarly to Quicksort. Let H_m denotes the m^{th} harmonic number. Knuth ([8] Exercise 5.2.2-32) showed that the expected number of comparisons to select the median of s elements (with $s = 2m - 1$) by this algorithm is

$$FIND(m) = 4m(H_{2m-1} - H_m) + 4m - 4H_m + \frac{4}{3}.$$

The last term disappears when $m = 1$. Asymptotically this function has value near $3.39s$.

Floyd and Rivest [2, 3] presented a selection algorithm requiring $3s/2 + O(\sqrt{s})$ comparisons to find the median in the expected case. Both FIND and SELECT can require a quadratic number of comparisons in the worst case: Schönhage, Paterson, and Pippenger [15] describe an algorithm that finds the median of s elements using $3s + o(s)$ comparisons in the worst case.

2 Analyses of Selection Strategies

We assume that the keys to be sorted are distinct, and that all initial input permutations are equally likely; for analysis purposes, this is equivalent to taking a random sample of the elements at each level. We assume throughout that the sample size is found by rounding the sampling function value to the nearest odd integer. We follow standard analyses in assuming that a sample is taken with replacement when $t > n$ and that partitioning cost at a single stage is equal to $n + 1$.

It will be convenient to distinguish between N, the input problem size, and n, the size of a subarray somewhere during the recursion. Thus we have $n = N$ only at the first stage of recursion.

Suppose that $\sigma(n)$ returns odd positive integers and let $h(n) = (\sigma(n) - 1)/2$. Let $f(s)$ denote the expected cost of selecting the median of s items. Then the total expected cost of Quicksort with sampling strategy $\sigma(n)$ is given by the following recurrence.

$$C_\sigma(n) = n + 1 + f(\sigma(n)) + 2\sum_{r=1}^{n} \frac{\binom{n-r}{h(n)}\binom{r-1}{h(n)}}{\binom{n}{\sigma(n)}}\left[C_\sigma(r-1)\right]$$

The summation term gives the probability that the element with rank r is the sample median: equivalently, that a sample of size $\sigma(n)$ is chosen so that $h(n)$ items have rank below r and $h(n)$ items have rank above r.

We would like to derive a closed form for arbitrary $\sigma(n)$ and find the $\sigma(n)$ that minimizes $C_{\sigma(n)}$; clearly this is quite difficult. Instead, we will find optimal functions for certain classes of strategies. We consider the following classes.

1. In *fixed-sample (FS) strategies* the sample size $\sigma(n)$ equals a constant t for all $n \in 1..N$. (When $n < t$ a sample of size t is drawn with replacement from the subarray.) Total costs for these strategies can be derived from Sedgewick's [13], [14] analyses.

2. In *two-tier (TT) strategies* the sample size is determined by a function of the form

   $$\sigma_t(n) = \begin{cases} \gamma(t,n) & \text{if } n = N \\ t & \text{if } n < N \end{cases}$$

   We derive the optimal formula $\gamma^*(t,n)$ for any t; it turns out that $\gamma^*(t,n) = \Theta(\sqrt{n})$ for any fixed t.

   When $\gamma^*(t,n)$ is applied, the "globally optimal" choice of t for given problem size N is denoted t^*. We show that the globally optimal TT strategy has lower cost than any FS strategy.

3. *Square-root (SQ) strategies* have sampling functions of the form $\sigma(n) = \Theta(\sqrt{n})$. We prove that the SQ strategy which applies

   $$\sigma(n) = \gamma^*(t^*, n) \qquad \text{for} \quad 1 \le n \le N$$

has lower total cost than the globally optimal TT strategy.

We also show that any SQ strategy has sub-quadratic total cost in the worst case.

2.1 Optimal TT Strategies

We first derive the optimal two tier strategy $\gamma^*(1,n)$ for $t = 1$. Sedgewick's [14] [13] analyses imply that, with $t = 1$, total expected partition cost for a problem of size n is $2(n+1)(H_{n+1} - 1)$ Selection cost is clearly zero for a one-element sample. Therefore the expected cost of a TT strategy with $t = 1$ is

$$TT_1(n) = n + 1 + f(2h+1) + \frac{2}{\binom{n}{s}} \sum_{r=1}^{n} \binom{n-r}{h}\binom{r-1}{h} 2r(H_r - 1).$$

Considering the summation term alone, we have

$$\sum_{r=1}^{n} \binom{n-r}{h}\binom{r-1}{h} 2rH_r - \sum_{r=1}^{n} \binom{n-r}{h}\binom{r-1}{h} 2r,$$

which is equivalent to

$$2(h+1)\sum_{r=1}^{n} \binom{n-r}{h}\binom{r}{h+1} H_r - 2(h+1)\sum_{r=1}^{n} \binom{n-r}{h}\binom{r}{h+1}.$$

Using identities found for example in [4], we obtain

$$2(h+1)\binom{n+1}{2h+2}\left(H_{n+1} - H_{2h+2} + H_{h+1} - 1\right),$$

and the complete formula for expected total cost is therefore

$$TT_1(n) = n + 1 + f(2h+1) + 2(n+1)\left(H_{n+1} - H_{2h+2} + H_{h+1} - 1\right).$$

Our goal is to find the value of h (and therefore of s) that minimizes $TT_1(n)$. Assume that a linear-expected-time selection algorithm is used, and replace $f(2h+1)$ by the first-term approximation $a(2h+1)$ for appropriate constant a. Replacing each harmonic number by the first three terms of the approximation $H_n \approx \ln n + \delta + 1/2n - O(1/n^2)$ (where $\delta \approx .577216$ is Euler's constant), we obtain

$$TT_1(n) \approx n + 1 + a(2h+1) + 2(n+1)\left(\ln(n+1) - \ln 2 + \delta + \frac{1}{2(n+1)} + \frac{1}{4(h+1)}\right).$$

This is minimized when

$$h^* = \sqrt{\frac{(n+1)}{4a}} - 1 \qquad \text{or} \qquad s^* = \gamma^*(1,n) = \sqrt{(n+1)/a} - 1.$$

Because of the approximation we have only found the first term of the optimal sampling function. This function must be rounded to the nearest odd integer to obtain a sample size: therefore the approximation is sufficient to give the true optimal sample size asymptotically, since with $\gamma^*(1,n) = \theta(\sqrt{n})$ the approximation is accurate to within an $O(1/n)$ term.

This analysis can be generalized to show that the optimal strategy for any t is a square-root function of n. Let $s = 2h + 1$ as before and, similarly, let $t = 2k + 1$. We have total partitioning cost

$$P_t(n) = \frac{(n+1)H_{n+1}}{H_{2k+2} - H_{k+1}} + O(n),$$

and total expected selection cost equal to $\Theta(nt)$ (see [14], [13]). Then

$$TT_t(n) = n + 1 + f(2h+1) + 2\sum_{r=1}^{n} \left[\frac{\binom{r-1}{h}\binom{n-r}{h}}{\binom{n}{2h+1}} \frac{rH_r}{H_{2k+2} - H_{k+1}} + \Theta(kr) \right]$$

which has closed form

$$TT_t(n) = n + 1 + f(2h+1) + (n+1)\left[\frac{H_{n+1} - H_{2h+2} + H_{h+1}}{H_{2k+2} - H_{k+1}} + \Theta(k) \right].$$

Optimal sample size is given by

$$h^* = \sqrt{\frac{n+1}{8a\,(H_{2k+2} - H_{k+1})}} - 1 \qquad \text{or} \qquad s^* = \gamma^*(t,n) = \sqrt{\frac{n+1}{2a\,(H_{2k+2} - H_{k+1})}} - 1. \quad (1)$$

The sum $H_{2k+2} - H_{k+1}$ is approximately equal to $\ln 2 - 1/(4k+4) \approx .693 - 1/(2t+2)$.

For any problem size N there must be some optimal t^* that minimizes total comparison cost over all TT strategies which use $\gamma^*(t,n)$. Our computational results suggest that optimal t^* is given by a slowly increasing function of N: for now we denote the optimal sampling function by

$$\sigma_N^*(n) = \begin{cases} \gamma^*(t^*,n) & \text{if } n = N \\ t^* & \text{if } n < N \end{cases}$$

Thus the optimal strategy is a "global" strategy for problem size N rather than one that only depends upon subarray size. The total cost of sorting an array of size n by this strategy is denoted by $TT_N^*(n)$.

2.2 The Square-Root Strategy

We now consider a *square-root strategy* which applies the sampling function

$$\sigma(n) = \gamma^*(t^*,n) \qquad \text{for} \quad 1 \le n \le N.$$

The $\gamma^*(t^*,n)$ and t^* here are as derived above for the optimal TT strategy.

We can rank the following five strategies.

- Let $\widehat{FS}(N)$ denote the cost of the optimal FS strategy which applies $\sigma(n) = \hat{t}$, where $\hat{t}$ is optimal for N. Optimal $\hat{t}$ grows slowly with problem size N (see [14], [13]).

- Let $FS^*(N)$ denote the expected cost of the FS strategy that applies $\sigma(n) = t^*$, where t^* is optimal for TT strategies. Note that t^* does not necessarily equal $\hat{t}$.

- Let $TT_N^*(N)$ denote the cost of the globally optimal TT strategy for problem size N, derived in the previous section.

- Let $\widehat{TT}_N(N)$ denote the cost of the TT strategy that applies

$$\sigma(n) = \begin{cases} \gamma^*(\hat{t}, n) & \text{if } n = N \\ \hat{t} & \text{if } n < N \end{cases}$$

  where $\hat{t}$ is as defined above.

- Let $SQ(n)$ denote the cost of the strategy which applies

$$\sigma(n) = \gamma^*(t^*, n) \qquad \text{for} \quad 1 \leq n \leq N.$$

With these definitions, we have the following theorem.

Theorem: *For a problem of size N, total comparison costs are related by*

$$SQ(N) \leq TT_N^*(N) \leq \widehat{TT}_N(N) \leq \widehat{FS}(N) \leq FS^*(N).$$

Proof: Note that the problem size N is fixed beforehand and determines the globally optimal t^* and $\hat{t}$ for their respective strategy classes. The theorem claims only that the relationships hold for the *total* expected cost of sorting, not necessarily for $n < N$.

We will demonstrate the inequalities working from right to left and numbering the arguments for reference.

(1) The fourth inequality is immediate by the definition of $\hat{t}$ as optimal for fixed-sample strategies at problem size N.

(2) The third inequality is immediate by the derivation of $\gamma^*(t, n)$ as optimal for any t. When $n < N$ the two strategies are identical. When $n = N$, the two-tier strategy choses a sample $\gamma^*(\hat{t}, n)$ which is optimal.

(3) The second inequality is immediate from the definition of t^* as optimal for all TT strategies with problem size N.

(4) We shall establish the first inequality by induction on problem size N. Let R be an arbitrary value smaller than N, and consider the TT_R^* strategy that applies

$$\sigma_R(n) = \begin{cases} \gamma^*(t^*, n) & \text{if } n = R \\ t^* & \text{if } n < R \end{cases}$$

The cost of sorting an array of size R by this strategy is denoted $TT_R^*(R)$. This is different from the TT_N^* strategy above because the "top level" for this strategy is at $n = R$ rather than at $n = N$. (The t^* is the same in both, however).

The induction hypothesis is that $SQ(R) \leq TT_R^*(R)$ for all $R < N$. Let $FS^*(R)$ be the cost of applying the FS strategy (with t^*) to a problem of size R. By the optimality of $\gamma^*(t^*, n)$ we have

$$SQ(R) \leq TT_R^*(R) \leq FS^*(R).$$

Now, $TT_N^*(R) = FS^*(R)$ whenever $R < N$, since the sampling functions are identical. Therefore we have $SQ(R) \leq TT_N^*(R)$ whenever $R < N$.

Consider now the costs $SQ(N)$ and $TT_N^*(N)$. When $n = N$, the two strategies are identical since they both take samples of size $\gamma^*(t^*, N)$. Letting $C(N)$ denote partition plus selection cost when $n = N$, and letting $P(N, r)$ denoting the probability that the partition element has rank r, the cost functions for the two strategies are

$$SQ(N) = C(N) + 2 \sum_{r=1}^{N} P(N,r)\, SQ(r)$$

and

$$TT_N^*(N) = C(N) + 2 \sum_{r=1}^{N} P(N,r)\, TT_N^*(r).$$

Since $SQ(r) \leq TT_N^*(r)$ for all $r < N$ the theorem is proved. As a basis case, note that γ^* will return 1, easily shown to be optimal for small n, when n is small. □

Argument (4) implies the that SQ strategy is better at all recursive stages.

Corollary: $SQ(n) \leq ST_N^*(n)$ for all $1 \leq n \leq N$.

Finally, it is easy to demonstrate subquadratic worst-case performance for any square-root strategy, even when the median-selection routine has quadratic worst-case cost.

Theorem: *The worst-case total cost of any square-root strategy is* $O(n^{1.5})$.

Proof: Suppose that a selection algorithm with worst-case cost $O(s^2)$ is used. To simplify notation, assume that the sample drawn is of size at least $2b\sqrt{n} + 1$, and at most $c\sqrt{n}$ for appropriate constants b and c. Worst-case partitioning cost occurs when extremal values of every subarray are chosen as partition elements. But if a sample of size at least $2b\sqrt{n} + 1$ is drawn at each level, it is impossible for elements with rank less than or equal to $b\sqrt{n}$ (or greater than $n - b\sqrt{n}$) to become sample medians. Therefore the worst-case cost for any SQ strategy is bounded by the following recurrence,

$$SQ(n) \leq n + 1 + (c\sqrt{n})^2 + SQ(b\sqrt{n}) + SQ(n - b\sqrt{n}),$$

which has closed form $O(n^{1.5})$. Note that this bound holds even if worst-case selection cost is quadratic in the sample size.

n	35	93	197	337	515	730	984	1274	1603	1968	2372
s_n	3	5	7	9	11	13	15	17	19	21	23
n	2813	3292	3808	4362	4953	5582	6248	6952	7693	8472	9289
s_n	25	27	29	31	33	35	37	39	41	43	45

Figure 1: Optimal Sample Size

3 An Exact Optimal Strategy

Given an exact formula $f(s)$ for selection cost it is possible to compute directly the optimal sample size by a dynamic-programming algorithm that finds $s = \sigma(n)$ to minimize

$$OPT(n) = n + 1 + f(s) + \sum_{r=1}^{n} \frac{\binom{r-1}{h}\binom{n-r}{h}}{\binom{n}{s_n}} [OPT(r-1) + OPT(n-r)].$$

The running time of the computation is just quadratic in the largest n examined, since it is only necessary to compare the current optimal value of s to $s+2$ at each value of n, rather than considering all possible values.

Hoare's selection algorithm FIND, although not the most efficient known, does permit an exact formula for selection cost, given in Section 1. Using this cost for $f(s)$ in the above computation of $OPT(n)$, the dynamic program produces the optimal sample sizes shown in Figure 1: for example, 5 is the optimal sample size for $93 \leq n < 197$. The computations were performed for n up to 10,000. Linear least-squares regression on a log-log scale suggests that the best power-law fit for this data is proportional to $n^{.482}$ (with residuals curving upwards); this is nearly a square-root strategy.

Table 2 compares total expected cost for the optimal strategy to fixed-sample strategies with $t = 1$ and $t = 3$. The numbers in parentheses give the ratio to the lower bound $N \lg(N) - N/\lg(N)$ (see [7]) on expected comparisons for sorting.

The fixed-sample costs $C_1(n)$ and $C_3(n)$ correspond to standard implementations found in [14] [13] and [8] (Section 5.2.2 and Exercises). These implementations are highly tuned to minimize running time in the MIX instruction set. The optimal strategy uses an inefficient general-purpose selection algorithm (Hoare's) and does not adopt some further improvements mentioned below. Nevertheless, the total comparison cost is less than either fixed strategy for N greater than 4000.

4 Remarks

Further limited experiments suggest that comparison cost is quite robust with respect to small variations in the sampling function; perhaps a hybrid strategy that choses one of a small set of sample sizes, each corresponding to a highly tuned subroutine, would give minimum execution time.

N	2000	4000	6000	8000
$C_opt(N)$	24664 (1.14)	53502 (1.13)	83879 (1.12)	115254 (1.12)
$C_3(N)$	24420 (1.12)	53581 (1.13)	84535 (1.13)	116654 (1.13)
$C_1(N)$	27395 (1.26)	60321 (1.27)	95339 (1.27)	131716 (1.28)

Figure 2: Total comparison costs for three strategies.

Since all median-selection algorithms manage to partition a sample around the median, a significant reduction in comparisons could be realized by avoiding re-examining the sample during the partitioning stage. The partitioning cost at each stage would be reduced from $n-1$ to $n-s$. If the sample region is contiguous within the subarray, sentinels can be used to eliminate boundary checks.

Finally we note that a strategy that minimizes total comparisons does not necessarily minimize total running time: the number of *exchanges* of items also contributes to the leading term of running time. Experimental comparisons of fixed-sample strategies and the optimal strategy for other cost measures are described in [11] and [12].

Acknowledgements

We thank Jon Bentley who first suggested the idea of allowing sample sizes to vary as a function of array size, and Mike Langston who suggested the hybrid scheme.

References

[1] C. R. Cook and D. J. Kim. Best sorting algorithm for nearly sorted lists. *Communications of the ACM*, 23(11):620–624, November 1980.

[2] R. W. Floyd and R. L. Rivest. The algorithm select—for finding the ith smallest of n elements. *Communications of the ACM*, 18(3):173, March 1975.

[3] R. W. Floyd and R. L. Rivest. Expected time bounds for selection. *Communications of the ACM*, 18(3):165–172, March 1975.

[4] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Welsey Publishing Company, Reading, MA, 1989.

[5] C. A. R. Hoare. Partition (algorithm 63), quicksort (algorithm 64), and find (algorithm 65). *Communications of the ACM*, 4(7):321–322, July 1961.

[6] C. A. R. Hoare. Quicksort. *Computer Journal*, 5(4):10–15, April 1962.

[7] D. E. Knuth. *The Art of Computer Programming: Volume 2, Seminumerical Algorithms.* Addison-Wesley Publishing Company, Reading, MA, 1973.

[8] D. E. Knuth. *The Art of Computer Programming: Volume 3, Sorting and Searching.* Addison-Wesley Publishing Company, Reading, MA, 1973.

[9] R. Loeser. Some performance tests of "quicksort" and "descendents.". *Communications of the ACM*, 17(3):143–152, March 1974.

[10] R. Loeser. Survey on algorithms 347, 426, and quicksort. *ACM Transactions on Mathematical Software*, 2(3):290–299, September 1976.

[11] C. C. McGeoch. *Experimental Analysis of Algorithms.* Ph.D. Dissertation, CMU-CS-87-124, Carnegie Mellon University, 1986.

[12] C. C. McGeoch. An experimental study of median-selection in quicksort. *Proceedings, 24th Allerton Conference on Communication, Control, and Computing, Univ. of Illinois*, 19–28, October 1986.

[13] R. Sedgewick. Analysis of quicksort programs. *Acta Informatica*, 7(4):327–355, 1977.

[14] R. Sedgewick. *Quicksort.* PhD thesis, Stanford University, 1975.

[15] A. Shonhage, M. Paterson, and N. Pippenger. Finding the median. *Journal of Computer and System Sciences*, 13:184–199, 1976.

[16] R. C. Singleton. An efficient algorithm for sorting with minimal storage (algorithm 347). *Communications of the ACM*, 12(3):185–186, March 1969.

A New Algorithm for Generating Necklaces

Terry MinYih Wang
Carla D. Savage*
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27695

July 12, 1990

Abstract

In this paper we present an algorithm for generating the lexicographically smallest representatives of all the n-bead necklaces in k-colors. The time required is $O(nN_k^n)$, where N_k^n is the number of necklaces with n beads of k colors. To our knowledge, this is the first algorithm for this problem which has been proved to asymptotically improve the obvious $O(n * k^n)$ approach of examining all k-ary n-tuples and rejecting those which are not lexicographically smallest rotations. Our algorithm has another advantage over a competitive one proposed by Fredericksen, Kessler, and Maiorana in that it can be implemented so that there is a gap of at most one non-necklace examined between any two necklaces generated, whereas the FKM algorithm can have gaps as large as $\lfloor (n-1)/2 \rfloor$. Although no good upper bound was established for FKM, we show experimentally that its performance appears to be as least as good as ours for $k > 2$.

1 Introduction

In this paper, we consider the problem of efficiently generating necklaces. A *necklace of n beads in k colors* is an equivalence class of k-ary n-tuples under rotation. That is, if σ denotes the rotation

$$\sigma(x_{n-1} \dots x_0) = x_{n-2} \dots x_0 x_{n-1}$$

---

*This work was supported in part by the National Science Foundation Grant No. CCR8906500.

then the k-ary n-tuples x and y are in the same equivalence class or *necklace* if and only if $\sigma^j(x) = y$ for some integer j.

Let N_k^n denote the number of n-bead necklaces in k colors. Since no necklace can contain more than n elements, and the necklaces a^n for $0 \leq a \leq k-1$ have only one element,

$$N_k^n \geq k + ((k^n) - k))/n$$

and equality holds when n is prime. So, in applications where every rotation of a given k-ary n-tuple represents the same phenomenon, it would be more efficient to work with only one representative of each necklace, rather than all k^n of the n-tuples. A necklace is thus identified with its representative.

A simple and elegant algorithm was proposed in [FrMa] and [FrKe] to generate for each necklace the lexicographically largest element. We will refer to this as the *FKM algorithm.* A disadvantage of the FKM algorithm is that there can be gaps in which as many as $\lfloor (n-1)/2 \rfloor$ non-necklaces are examined between any two necklaces generated. For the case $k = 2$ the authors of [FrKe] prove an upper bound of $\lfloor (n-1)/2 \rfloor N_k^n$ on the number of n-tuples generated and examined by the algorithm. The time spent generating and checking each element is $O(n)$, giving an overall time for the algorithm of $O(n^2 N_k^n)$. However, this does not establish an asymptotic improvement over the direct approach of generating *every* k-ary n-tuple and checking each one to see if it is the representative of its necklace, at a cost of time $O(n)$ per check.

In this paper, we describe a new algorithm which generates the lexicographically smallest element in each necklace. The number of k-ary n-tuples generated and examined by our algorithm is at most *twice* the total number of necklaces, at a cost of $O(n)$ per n-tuple. This gives a total time of $O(nN_k^n)$. Thus, in contrast to the FKM algorithm, we can prove our algorithm gives an asymptotic improvement over the obvious approach of testing in linear time every k-ary n-tuple. Our algorithm also has the advantage that it can be implemented so that there is a gap of at most one non-necklace examined between any two necklaces generated.

We also present the results of our experiments comparing the actual performance of our algorithm against the FKM algorithm. The indication is that (1) our algorithm is better than FKM for $k = 2$, (2) FKM is at least comparable to our algorithm for $k > 2$ and (3) the upper bound on FKM from [FrKe] is almost certainly not tight.

In Section 2 we describe our basic algorithm for the case $k = 2$, prove that it examines no more than $2N_k^n$ of the n-tuples, discuss how to implement it to minimize gaps, and compare it to FKM. In Section 3 we extend to arbitrary k and prove that it is still possible to stay within the $2N_k^n$ bound.

Most proofs have been omitted in this extended abstract.

2 Necklaces of Beads in Two Colors

Our idea for generating necklaces of two-color beads was inspired by a result in [LiHiCa] that a certain variation on the shuffle-exchange graph is Hamiltonian.

When $k = 2$, the n-tuples are bit strings which we regard as elements of $\{0,1\}^n$. Let $\alpha\beta$ denote the concatenation of bitstrings α and β and let 0^t denote the bitstring of length t in which every symbol is 0.

Define a function τ on $\{0,1\}^n$ by

$$\tau(x) = \begin{cases} \alpha 0 & \text{if } x = \alpha 1 \text{ for some } \alpha \in \Sigma^{n-1} \\ \alpha 1 & \text{if } x = \alpha 0 \text{ for some } \alpha \in \Sigma^{n-1} \end{cases}$$

As in the previous section, $\sigma(x)$ denotes the rotation of string x one position left.

We choose the *lexicographically smallest* string as the representative of a necklace and refer to this string itself as a necklace. A tree of necklaces is generated as follows. Starting with the string $x = 0^n$ as root, we generate as the children of x all those necklaces of the form $\tau\sigma^j(x)$ for some j satisfying $1 \leq j \leq n-1$. This procedure is applied recursively to each child of x. We show first that every necklace will be generated.

Lemma 1 *Every n-bead necklace will appear in the tree with root 0^n.*

Proof. Let x be an n-bead necklace. We show by induction on t, the number of ones in x, that x is in the tree. If $t = 0$ then $x = 0^n$, which is the root of the tree. For $t > 0$, note that $x = \alpha 1$ for some $\alpha \in \{0,1\}^{n-1}$ since x must be the lexicographically smallest in its class. Thus $x = \tau\sigma(0\alpha)$. The representative y of the class containing 0α has $t-1$ ones, so must be in the tree, by induction. Since 0α is a rotation of y, $x = \tau\sigma^j(y)$, for some j, so x will be a child of y in the tree. □

Although at first glance it would appear that we need to examine *every* string of the form $\alpha 1$ with this procedure, this is not the case. We show in Theorem 1 below that if x, $\tau(x)$, and $\sigma(x)$ are not necklaces, then $\tau\sigma(x)$ is not a necklace.

It follows that for a *necklace* $y \neq 00\ldots01$, that if $\tau\sigma^j(y)$ is *not* a necklace for some j, then $\sigma^{j+1}(y) = y$, $\tau\sigma^{j+1}(y)$ is also *not a necklace.*

As a result, from a necklace y, we generate the $\tau\sigma^j(y)$, starting with $j = 1$, until the first j is found for which $\tau\sigma^j(y)$ is not a necklace. By Theorem 1, we are guaranteed that at this point all children of y have been found.

Lemma 2 *If* $x = 0^t 1\alpha 1$, *where* $t > 0$ *and* $\alpha \in \{0,1\}^*$, *then*

(a) if there are $\beta, \gamma \in \{0,1\}^*$ *such that* $\alpha = \beta 0^{t+1}\gamma$ *then* x *is not a necklace.*

(b) if x *is not a necklace, then there are* $\beta, \gamma \in \{0,1\}^*$ *such that* $\alpha = \beta 0^t \gamma$.

Theorem 1 *For bitstring* $x \in \{0,1\}^n$, *if* x, $\tau(x)$, *and* $\sigma(x)$ *are not necklaces, then* $\tau\sigma(x)$ *is not a necklace.*

We summarize below the recursive algorithm for generating the lexicographically smallest representatives of the n-bead necklaces in two colors. Note that by Theorem 1 and as discussed above, while generating the children of the necklace y in the tree, at most one n-tuple is examined which is *not* a necklace. Thus the total number of n-tuples examined is at most $2N_2^k$.

This algorithm makes use of the fact that a necklace of the form $y = 1\alpha$ will not have any children.

Two Color Algorithm

```
procedure search(y);
        begin
                output(y);
                done := false;
                while (not(done) and
                    (the first symbol of y is 0)) do begin
                        y := sigma(y);
                        x := tau(y);
                        if x is a necklace then search(x)
                                else done := true
                        end
        end;

main program
        begin
        y := 00...0;
        output(y);
        y := tau(y);
        search(y)
    end.
```

A recursive implementation will give rise to *gaps* in this algorithm: as necklaces are implicitly stacked, it could occur that for several consecutive necklaces on the stack, the next candidate "child" to be tested turns out to be a non-necklace. We can avoid this gap with a nonrecursive implementation in which a necklace y is pushed on the stack only after checking that the next n-tuple x to be examined when y is popped is actually a necklace. In this case, x can be saved on the stack with y to avoid generating and checking x twice.

Finally, note that in the two color case 01^{n-1} is the only necklace of the form 01α which has a child in the tree. Thus we could search in the Two Color algorithm for children of y only when y has the form 00α and then output 1^n at the end of the main procedure. This helps for small values of n by delaying the asymptotic behavior.

With this modification incorporated, our algorithm is compared with FKM in the table of Figure 1. The ratio of non-necklaces examined by the algorithms to the total number of necklaces is compared.

3 Necklaces of Beads in k Colors

In the general case of generating necklaces of beads with k colors, let $\Sigma = \{0, 1, \ldots k-1\}$. For each $a \in \Sigma \setminus \{0\}$, define a function τ_a on elements of Σ^n by

$$\tau_a(x) = \begin{cases} \alpha a & \text{if } x = \alpha 0 \text{ for some } \alpha \in \Sigma^{n-1} \\ \text{undefined} & \text{otherwise} \end{cases}$$

We propose to generate a tree of all n-bead k-color necklaces as follows. Starting from the root $y = 0^n$, generate as children of y all *necklaces* of the form $\tau_a \sigma^t(y)$ for $a \in \Sigma \setminus \{0\}$, $1 \le t \le n-1$. This procedure is then applied recursively to the children of y. We can prove as in Lemma 1 that all necklaces will be in the tree with root 0^n. We focus now on minimizing the number of non-necklaces examined when generating the children of a necklace y.

Lemma 2 of Section 2 is still valid and can be restated as follows.

Lemma 3 *If $x = 0^t a\alpha b$ where $t > 0, \alpha \in \Sigma^*$, and $a, b \in \Sigma \setminus \{0\}$ then*

n	N_2^n	B/N for FKM	B/N for Our Algorithm
1	2	0	0
2	3	0	0
3	4	.250	0
4	6	.333	0
5	8	.750	.250
6	14	.643	.286
7	20	1.050	.500
8	36	.972	.556
9	60	1.117	.667
10	108	1.093	.722
11	188	1.191	.798
12	352	1.122	.824
13	632	1.179	.870
14	1182	1.147	.892
15	2192	1.153	.914
16	4116	1.138	.931
17	7712	1.141	.945
18	14602	1.126	.955
19	27596	1.125	.964
20	52488	1.115	.971
21	99880	1.111	.977
22	190746	1.104	.981
23	765768	1.100	.985
24	699252	1.195	.988
25	1342184	1.092	.990

Figure 1: Comparison of our algorithm to FKM for $k = 2$. (N is the number of necklaces and B is the number of non-necklaces examined.)

(a) if there are $\beta, \gamma \in \Sigma^*$ *such that* $\alpha = \beta 0^{t+1} \gamma$ *then* x *is not a necklace.*

(b) if x *is not a necklace, then there are* $\beta, \gamma \in \Sigma^*$ *such that* $\alpha = \beta 0^t \gamma$.

We now show that a variation on Theorem 1 holds for k colors.

Theorem 2 *If* $x \in \Sigma^n$ *is not a necklace and if* $\tau_a(x)$, $a \in \Sigma \setminus \{0\}$, *is either (i) undefined or (ii) is not a necklace, then* $\tau_a \sigma(x)$ *is either undefined or is not a necklace.*

By Theorem 2, when generating the children of necklace y, one need only examine, for each $a \in \Sigma \setminus \{0\}$, the strings

$$\tau_a \sigma^1(y), \tau_a \sigma^2(y), \ldots$$

until the smallest t is found for which $\tau_a \sigma^t(y)$ is undefined or is not a necklace. This would imply at most $k-1$ non-necklaces need be examined when generating the children of y.

However, we can do better. At necklace y, for each j satisfying $1 \leq j \leq n-1$, let $s(j)$ be the largest $l \leq n-1$ for which $\tau_l \sigma^j(y)$ is either not defined or is not a necklace. Theorem 2 shows that

$$s(1) \leq s(2) \leq \ldots$$

but in fact we have the following.

Lemma 4 *If* $s(j) > 0$ *for some* j, *then* $s(j+1) = k-1$.

Finally, we make use of the following lemma, a variation of which appears in [FrMa].

Lemma 5 *For* $x = 0\alpha \in \Sigma^n$, *and* $a \in \Sigma \setminus \{0\}$, *if* $\tau_a \sigma(x)$ *is not a necklace, the neither is* $\tau_{a-1} \sigma(x)$.

Putting these results together we have the following.

Theorem 3 *Let the candidate children of necklace y be generated in the order:*

$$\begin{gathered}\tau_{k-1}\sigma^1(y), \tau_{k-2}\sigma^1(y), \ldots, \tau_{s(1)}\sigma^1(y),\\ \tau_{k-1}\sigma^2(y), \tau_{k-2}\sigma^2(y), \ldots, \tau_{s(2)}\sigma^2(y),\\ \vdots\end{gathered}$$

Then all children of y have been generated by the time we examine the first element which is undefined or not a necklace.

Theorem 2 establishes that no more than $2N_k^n$ of the n-tuples are generated and examined while growing the tree of necklaces.

As in the two color case, we can implement this algorithm nonrecursively in such a way that no more than one non-necklace will be examined between any two necklaces. We can again postpone the asymptotic behavior by generating children of necklace y only if y is of the form $y = 00\alpha$; necklaces with no '0' beads can be generated recursively as $k-1$-color necklaces. Our algorithm, with this modification incorporated, is compared with FKM in Figure 2.

4 Final Remarks

Two obvious open questions remain. There should be a way to prove that the FKM algorithm actually examines at most cN_k^n of the n-tuples for some constant c. Extrapolating the results in Figure 2, it seems that for $k > 2$ the FKM algorithm may always have B/N smaller than our algorithm. It would be nice to be able to prove this.

More important, is there a way to generate all necklaces in total time $O(N_k^n)$, ignoring the time for output? There are several combinatorial classes for which such a result is possible using Gray code - like algorithms ([Gi], [Jo], [Lu], [Ru], [Sa]). Alternatively, some clever encoding scheme, taking advantage of the structure of the tree, may allow one to reduce or eliminate the $O(n)$ cost of necklace testing in our algorithm.

References

[FrKe] H. Fredericksen and I. J. Kessler, "An algorithm for generating necklaces of beads in two colors," *Discrete Mathematics* 61 (1986) 181-188.

[FrMa] H. Fredericksen and J. Maiorana, "Necklaces of beads in k colors and k-ary de Bruijn sequences," *Discrete Mathematics* 23, No. 3 (1978) 207-210.

[Gi] E. N. Gilbert, "Gray codes and paths on the n-cube," *Bell Systems Technical Journal* (1958) 815-826.

[Jo] S. M. Johnson, "Generation of permutations by adjacent transpositions," *Math. Comp.* 17 (1963) 282-285.

n	k	N_2^n	B/N for FKM	B/N for Our Algorithm
3	3	11	.273	0
4	3	24	.333	.042
5	3	51	.569	.176
6	3	130	.508	.231
7	3	315	.613	.308
8	3	834	.580	.360
9	3	2195	.595	.413
10	3	5934	.581	.460
11	3	16107	.582	.506
12	3	44368	.571	.547
13	3	122643	.568	.586
14	3	341802	.562	.621
15	3	956635	.558	.655
16	3	2690844	.554	.684
4	4	70	.286	.043
5	4	208	.413	.115
6	4	700	.377	.153
7	4	2344	.410	.190
8	4	8230	.393	.223
9	4	29144	.393	.255
10	4	104968	.385	.287
11	4	381304	.381	.318
5	5	629	.318	.097
6	5	2635	.294	.105
7	5	11165	.305	.127
8	5	48915	.294	.149
9	5	217045	.292	.170
6	6	7826	.235	.075
7	6	39996	.242	.091
8	6	210126	.235	.105
7	7	117655	.201	.068
8	7	720916	.195	.079

Figure 2: Comparison of our algorithm to FKM for $k > 2$. (N is the number of necklaces and B is the number of non-necklaces examined.)

[LiHiCa] W. Liu, T. H. Hildebrandt, and R. Cavin III, "Hamiltonian cycles in the shuffle-exchange network," *IEEE Transactions on Computers*, to appear.

[Lu] J. Lucas, "The rotation graph of binary trees is Hamiltonian," *Journal of Algorithms* 8 (1987) 503-535.

[Ru1] F. Ruskey, "Adjacent interchange generation of combinations," *Journal of Algorithms* 9 (1988) 162-180.

[Sa1] C. Savage, "Gray code sequences of partitions," *Journal of Algorithms* 10, No. 4 (1989).

Stacks, Queues, and Deques With Order-Statistic Operations

by

D.S. Richards, J.S. Salowe

Department of Computer Science
University of Virginia
Charlottesville, Virginia 22903

ABSTRACT

An *order-statistic deque* is a deque that also supports the operation *findkth*, where k is a constant. We show how to implement an order-statistic deque so that the amortized time for operations *inject*, *eject*, *push*, and *pop* are $O(\log k)$, and operation *findkth* can be performed in constant time. This deque is applied to two problems. In the first problem, let L be a list of n items, each item possessing a real-valued key, over which we run a window of width w. We show that the worst-case time complexity of this problem, the running k order-statistic problem, is $\Theta(n + (n-w) \log k)$. In the second problem, we show that, given a set S of n points in the plane, the smallest-area convex quadrilateral determined by S can be computed in $O(n^3)$ time.

1. Introduction

There are several applications where order-statistics need to be computed on a dynamically changing set. In this paper we give results for those situations where the changes are characterized by deque operations. We present algorithms, matching lower bounds, and examples of their applications. A *deque* is a data structure d that supports the following four operations on a sequence of elements:

inject(x,d): Insert x as the new last element of d.

eject(d): Delete the last item from d and return it.

push(x,d): Insert x as the new first element of d.

pop(d): Delete the first item from d and return it.

A *queue* supports *inject* and *pop* (or *push* and *eject*), and a *stack* supports *push* and *pop* (or *inject* and *eject*). An order-statistic stack, queue, or deque also supports *findkth*:

findkth(d): Returns k^{th} smallest element on d.

We show how to implement order-statistic stacks, queues, and deques so that *findkth* can be performed in constant time, and all other operations take $O(\log k)$ time. The time bounds for queues and deques are amortized.

Our algorithm is a generalization of that of Gajewska and Tarjan [GAJE86], who present a deque which supports *findmin* and for which all operations run in $O(1)$ amortized time; they also present an $O(1)$ worst-case time implementation. In related work, Atkinson, Sack, Santoro, and Strothotte present a structure called the order-statistics tree which, given a fixed collection of ranks, supports constant-time rank queries and $O(\log n)$-time insertions and deletions [ATKI86].

We apply our data structures, in particular the order-statistic queue, to two problems. The first problem is to implement one-dimensional order-statistic filters [MARA87], and we present an optimal-time algorithm. The second problem is to compute the smallest-area convex quadrilateral determined by a set of n points in the plane. We present a simple $O(n^3)$ time algorithm for this problem.

The structure of this paper is the following. Section 2 contains a lower bound on the amortized computational complexity of order-statistic stacks, queues, and deques. In Section 3, we present the three order-statistic structures, and in Section 4, we apply the order-statistic queue to the problem of computing running k order-statistics. Section 5 presents an algorithm to compute the smallest-area convex quadrilateral. In the last section, we make some remarks concerning other applications of our order-statistic structures.

2. Lower Bound

In this paper, we are concerned with the amortized computational complexity of maintaining order-statistic stacks, queues, and deques. We assume the algebraic decision tree model of computation, and we use the following lemma to justify our claims of optimality:

Lemma 1: Let S be a structure supporting *push* (x,S) and *findkth* (S). There is a sequence of $O(n)$ *push* (x,S) and *findkth* (S) operations which takes $\Omega(n \log k)$ time.

Proof: We first show that a sequence of $2k-1$ *push* and k *findkth* operations can be used to sort a list A of size k. Given A, we push it onto S, remembering the smallest key s. Then, for k iterations, we alternately perform *findkth* (S) and *push* (x,S), where the key of x is $s-1$; the output consists of A in nonincreasing order.

Given a collection of n/k lists $\{A_i\}$, each of size k, to sort, we perform a similar process. Assume for the ease of exposition that k divides n. Let $a_0 = \min \{ x \mid x \in A_{n/k} \}$, and let $a_i = \max \{ x \mid x \in A_i \}$. Form a list L, where $L[j+k(i-1)] = A_i[j] + \sum_{l=i+1}^{n/k} a_l$, $1 \le j \le k$, and $L[j] = a_0 - 1$, $n+1 \le j \le n+k-1$. The list L can be regarded as all of the lists A_i concatenated together, after augmenting each list so that each element A_i is greater than every element of A_{i+1}, with a dummy block of small values at the end. To process L, its elements are consecutively pushed onto S, with a *findkth* after each *push* (after the initial k elements of L have been pushed). Each list A_i is then produced in sorted order as in the procedure described above, by inserting the elements of A_{i+1}. Since of these n/k lists require $\Omega(k \log k)$ time, the result follows. ☐

3. Order-Statistic Structures

In this section, we show how to implement optimal (amortized) time order-statistic stacks, queues, and deques. We first show how to implement an order-statistic stack; the order statistic stack is a crucial component in the order-statistic deque. We then show how to implement an order-statistic queue. This queue appears to be more practical than the order-statistic deque for the application in Section 4. Finally, we present the order-statistic deque.

3.1. Preliminaries

In our structures below, we use a variety of trees. A *min heap* is generally represented by a (min) heap-ordered tree. (A *max heap* is similarly defined.) Let v be a node; denote the parent of v by $p(v)$ if $p(v)$ exists. A tree is said to have (min) heap order if for every non-root node v, $key(p(v)) \leq key(v)$. If a tree has heap order, an item with minimum key is at the root. There are a wide variety of heap implementations; the most well-known heap, the *2-heap*, was developed by Williams [WILL64]. This heap has constant-time *findmin* and $O(\log n)$ worst-case insertion and deletion times.

It is also possible to represent a heap using a binary search tree. A binary search tree is a rooted, directed binary tree whose nodes are in symmetric order: if node x contains an item with key t, every item in the left subtree of x has key less than or equal to t, and every item in the right subtree of x has key greater than t. For a binary search tree to support search for order-statistics, each node x is augmented with the number of nodes in its left subtree. It is possible to implement a "balanced" binary search tree having n nodes so that insertions, deletions, and searching for order-statistics takes $O(\log n)$ time.

We now define *Cartesian trees* [VUIL80]. Let A be a list of n items, $A[i]$ the item in position i, and $A[i:j]$ the contiguous sublist of A from position i to position j. If $n = 0$, then the Cartesian tree $T(A)$ of A is null. If $n > 0$, the root r of a Cartesian tree $T(A)$ is an item with smallest key, say $A[i]$. The left child of r is the root of a Cartesian tree on $A[1:i-1]$, and the right child of r is the root of a Cartesian tree on $A[i+1:w]$. It is easy to verify that a Cartesian tree is heap-ordered, and that an inorder traversal of a Cartesian tree gives A in left-to-right order. Cartesian trees support queue operations in constant amortized time [HARE84].

3.2. Order-Statistic Stacks

Let S be the name of the order-statistic stack. Suppose, without loss of generality, that all keys stored in S are distinct (if this is not the case, we can break ties with time-stamps). The order-statistic stack consists of three structures — two ordinary stacks, A and E, and a max heap H. All elements on the order-statistic stack are stored in A, and a subset of these elements are stored in E and H. Let y be the item with largest-valued key in H. To perform *push*(x,S), x is pushed onto A. If there are fewer than k items in H, x is inserted into H. Otherwise, if H contains k items, and if $key(x) < key(y)$, then x is inserted into H, y is removed, and y is pushed onto E. To perform *pop*(S) removing item x from S, x is removed from each structure it appears in (it turns out that x cannot be in E). In particular, if x appears in H, it is removed, E is popped, and the popped item is inserted into H. To perform *findkth*(S), we run *findmax*(H), provided H has size k. If H has size less than k, then *findkth*(S) returns *null*.

The time complexity of *push*(x,S) and *pop*(S) is $O(\log k)$ since the most expensive operation is the update of the max heap H, a heap of size at most k. The time complexity of *findkth*(S) is $O(1)$ provided the 2-heap is used to implement H.

Correctness relies on stack order. If $key(x)$ is larger than the keys of k items $x_1, \ldots, x_k$ already on the stack, then x is not a candidate for k^{th} smallest item; $x_1, \ldots, x_k$ are on S for as long as x is. Therefore x is placed into H if and only if $key(x)$ is one of the k smallest keys currently on the stack. The operations on the order-statistic stack maintain the following two invariants. First, if H contains k items, then the items in H have the smallest k keys currently on the stack. Second, E is a list in increasing order, from the top of the stack to the bottom, and all of the keys in E are larger than the largest key in H.

Let y be the item in H with largest key, and let $top(E)$ be the item on the top of stack E. To show the invariants are maintained, first we analyze $push(x,S)$. Note that when x is pushed onto S and $key(x) < key(y)$, x is added to H and y is removed from H. Since $key(x) < key(y)$, H contains the smallest k keys after the push. Further, since $key(top(E)) > key(y)$ prior to y's removal from H, E remains in increasing order. Now we analyze $pop(S)$, removing x from S. If x is popped and it was not on E or H, it could not have been possessed one of the k smallest keys, so the invariants still hold. If x is in E, this implies it does not have one of the k smallest keys and should not have been placed into H; this case is therefore impossible. If x is in H, it is removed, and the item on top of stack E is inserted into H. Stack E is still sorted. Further, any item q not in E or H when x is popped was not in H at the time x was pushed, so q cannot be one of the k smallest items currently in S.

The second invariant is used to prove the first invariant, and the first invariant immediately implies correctness. This proves the following theorem:

Theorem 1: An order-statistic stack S can be implemented so that $findkth(S)$ takes $O(1)$ time, and $push(x,S)$ and $pop(S)$ takes $O(\log k)$ time.

3.3. Order-Statistic Queues

To implement an order-statistic queue Q, we modify an order-statistic heap. An *order-statistic heap* h is a structure which supports $insert(x,h)$, $delete(x,h)$, and $findkth(h)$. An order-statistic heap is implemented with two heaps, one a max heap h_1 and the other a min heap h_2. There are exactly k items in h_1, each with key value less than or equal to the k^{th} smallest key. The remaining items are in h_2, and they have key values greater than or equal to the k^{th} smallest key. Procedure $findkth(h)$ returns the root of h_1. Procedures $insert(x,h)$ and $delete(x,h)$ cause items to be moved from h_1 to h_2 and vice versa. For instance, say an item with key value smaller than the k^{th} smallest key is inserted. This item is placed into h_1. Since the size of h_1 is now $k+1$, the key with largest value is removed from h_1 and inserted into h_2.

Order-statistic heaps must be modified for use in order-statistic queues, otherwise they would have $O(\log n)$ insertion and deletion times. We implement an order-statistic queue Q with two structures, an order-statistic heap M and a Cartesian tree C. M consists of 2 heaps, h_1 and h_2, and will contain at most $2k + 1$ items. Provided there are at least k items in the queue, heap h_1 is of size k and contains the items in Q with the smallest k keys; heap h_2 is of size at most $k+1$, and it is guaranteed to contain an item with $k+1^{th}$ smallest key in Q. Cartesian tree C will be used to determine the members of M. We assume that the items with k smallest keys in C are marked, and we maintain the invariant that nonmarked children of marked nodes are in h_2.

Let y be the item currently in Q with k^{th} smallest key. To implement $inject(x,Q)$, x is inserted into C. If $key(x) < key(y)$, then x is marked in C and inserted into h_1. Item y is removed from h_1 and placed into h_2; y becomes unmarked and all of its children in C are deleted from h_2. If $key(x) > key(y)$, then x is only inserted into h_2 if $p(x)$ is marked. If this is the case, any child of x in C, if there is one, is removed from h_2. To implement *pop* (Q), let x be the item popped. If x is not in M, it is simply deleted from C. If x is in h_1, x is deleted from both h_1 and C, then z, the item with smallest key in h_2, is marked, removed from h_2, and inserted into h_1. Finally, any children of z are inserted into h_2. If x is in h_2, it is deleted from both h_2 and C, and any child of x in C is inserted into h_2. (Note that queue-order deletions in the Cartesian tree imply that x, in bothcases above, has at most one child in C.) To implement *findkth* (Q), we run *findkth* (M).

The time complexities must be amortized since insertions and deletions are made into the Cartesian tree C. Assuming that h_2 contains no more than $k+1$ items, then except for updates to C, *inject* and *pop* take $O(\log k)$ time, and *findkth* takes $O(1)$ time if a 2-heap is used to implement the max heap. Since queue-order insertions and deletions into a Cartesian tree can be done in amortized constant time, the amortized time complexities for *inject* and *pop* are $O(\log k)$. Operation *findkth* can be performed in constant time.

Correctness of this algorithm relies on the structure imposed by the Cartesian tree. As an invariant, we maintain that the items with the k smallest keys are marked in C; these are the items in h_1. Define the frontier $F(T)$ of a of nonempty set T in C to be the set of nodes x in C such that x is not in T, but $p(x)$ is. If T is empty, $F(T)=\{r\}$, where r is the root of C. Consider the items in h_1, and let $F(h_1)$ be the frontier defined by these items. We maintain as a second invariant that the items in h_2 are precisely those in $F(h_1)$. It is easy to check that these invariants are maintained for the *inject* and *pop* operations.

Since C is a binary, (min) heap-ordered tree and the items in h_1 form a subtree of C, the size of $F(h_1)$ is at most $k+1$. Further, $F(h_1)$ contains the item in Q with $k+1^{th}$ smallest key, as this item must be a child of an item in h_1.

We summarize with the following theorem:

Theorem 2: There is a structure Q which implements *findkth* (Q) in constant time and implements *inject* (x,Q) and *pop* (Q) in amortized $O(\log k)$ time.

3.4. Order-Statistic Deques

Our order-statistic deque D consists of two order-statistic stacks, S_1 and S_2. Recall that an order-statistic stack S_i consists of three structures, A_i, E_i, and H_i. Besides the two order-statistic stacks, we also have a normal G (implemented as, say, a circular array) and pointers from G to corresponding items in H_1 and H_2. There is also an additional operation, *convert* (D), which is described below.

Initially, and until the size of the deque D reaches $2k$, D is implemented with two structures, a deque G and a binary search tree T. The details are omitted. When the size of the deque reaches $2k$, *convert* (D) is run. Operation *convert* (D) destroys all of D except G, and, using G, reconstructs S_1 and S_2 so that they are approximately the same size. Specifically, the midpoint of G is found, and a series of *inject* and *push* operations are performed on the elements in G. The items to the left of the midpoint are pushed onto S_1, and the items to the right of the midpoint are pushed onto S_2, each sequence in order from midpoint to endpoint. Throughout this conversion, a pointer to the item with k^{th} smallest key is maintained. This item t appears in one of the two heaps, say H_1. In H_2, a pointer indicates the item with largest key in H_2 less than $key(t)$; if no such item exists, the pointer is null.

To implement $inject(x,D)$, $push(x,S_2)$ is performed; to implement $push(x,D)$, $push(x,S_1)$ is performed. In either case, the two pointers must be adjusted so that the item with k^{th} smallest key can be found in one step. There are several cases, some of which involve the search for a predecessor or successor with respect to the sorted order. This requirement makes a binary search tree a better choice than a heap-ordered tree to implement H_1 and H_2, since search for an inorder predecessor or successor may be performed in time proportional to the depth of the tree. The details are omitted.

The *pop* and *eject* operations are more complicated, since the pop may have reduced the size of S_1 or S_2 to zero. To perform $pop(D)$, if $|S_1| \geq 1$, we perform $pop(S_1)$ and adjust the two pointers. If $|S_1| = 0$, we perform $convert(D)$ and then $pop(S_1)$. Operation *eject* is symmetric. Finally, operation $findkth(D)$ returns the contents of the appropriate pointer.

Since *convert* may be expensive, the time complexity of each operation is amortized. Define the amortized time a_i of the i^{th} operation to be $a_i = t_i + \phi' - \phi$, where t_i is the actual time, ϕ' is the potential after the operation, and ϕ is the potential before the operation. Let the potential ϕ of deque D be $\Big| \, |S_1| - |S_2| \, \Big| \log k$. Then, using Theorem 1, the amortized cost of $findkth(D)$ is $O(1)$, and the amortized cost of $push(x,D)$ or $inject(x,D)$ is $O(\log k)$, provided a balanced binary search tree is used to implement H_1 and H_2. If $convert(D)$ is not performed, the amortized cost of $pop(D)$ or $eject(D)$ is $O(\log k)$. The amortized cost of $convert(D)$ is at most

$$|D|\log k + \log k - |D|\log k = \log k.$$

This proves the following theorem:

Theorem 3: There is a structure D which implements $findkth(D)$ in constant time and implements $inject(x,D)$, $eject(D)$, $push(x,D)$ and $pop(D)$ in amortized $O(\log k)$ time.

4. Running k^{th} Order-Statistics

We now apply the order-statistic queue to the problem of computing running k^{th} order-statistics: Given a list L of n items, each item possessing a real-valued key, a window size w, and a rank k, $k \leq w/2$, the running k^{th} order statistic problem is to compute the k^{th} order-statistic in each of $L[1{:}w]$, $L[2{:}w+1]$, $\cdots$, $L[n-w+1{:}n]$. In the speech and image processing community, running k order-statistics are called order-statistic filters; order-statistic filters are used to enhance speech data and in image processing (e.g., [MARA87]; see [RICH90] for additional references). We give tight bounds on $C(n,w,k)$, the worst-case time complexity of computing running k order-statistics.

4.1. Lower Bounds for Running *k* Order-Statistics

In this subsection, we show that $C(n,w,k) = \Omega(n + (n-w)\log k)$. The proof technique is similar to that used in Section 2, and is related the techniques of Shamos [SHAM76] and Fredman and Spencer [FRED87].

Lemma 2: $C(w+k,w,k) = \Omega(w + k \log k)$.

Proof: Reduction from sorting. Given a list S of size k to sort, find the minimum and maximum items of S, a and b, respectively. Form a new list L, $L[i] = a-1$, $1 \le i \le k-1$, $L[k{:}2k-1] = S$, and $L[i] = b+1$, $2k \le i \le w+k$. Consider the output of a running k order-statistics algorithm; for $L[1{:}w]$, the output is the smallest item in $L[k{:}2k-1]$, for $[2{:}w+1]$, the output is the second smallest item in $L[k{:}2k-1]$, etc. Since any algorithm solving the running k order-statistic problem on a list of size $w+k$ must also look at every item, this proves the lower bound. ☐

Theorem 4: $C(n,w,k) = \Omega(n + (n-w)\log k)$.

Proof: Assume for ease of exposition that $n-w$ is a multiple of k. Consider a collection of $(n-w)/k$ lists of positive keys A_1, A_2, etc., each of size k. In order to sort all of these lists, $\Omega((n-w)\log k)$ time is required. Let $a_0 = \min\ \{x \mid x \in A_1\}$, and let $a_i = \max\ \{x \mid x \in A_i\}$. Form a list L, where $L[j] = a_0 - 1$, $1 \le j \le k$, $L[j+ki] = A_i[j] + \sum_{l=1}^{i-1} a_l$, and $L[j] = \sum_{l=1}^{(n-w)/k} a_l$, $n-w+k+1 \le j \le n$. The technique used in Lemma 2 can be applied to each A_i separately and independently, proving the theorem. ☐

4.2. An Algorithm

The order-statistic queue is the most practical of our order-statistic structures to compute running k order-statistics. Queue Q is initialized with the first w items in L. To do this, the Cartesian tree is constructed, the k^{th} smallest item in $L[1{:}w]$ is selected, appropriate nodes in C are marked, and M is built. Then, for i increasing from $w+1$ to n, a sequence of $n-w$ operations *findkth*(Q), *inject*$(L[i],Q)$, and *pop*$(L[i-w],Q)$ are performed. Initializing Q takes $O(w)$ time; by Theorem 1, *inject*(x,Q) and *pop*(Q) take amortized $O(\log k)$ time and *findkth*(Q) takes $O(1)$ time. Therefore, the entire algorithm takes $O(n + (n-w)\log k)$ time. With the lower bound from Theorem 4, this proves:

Theorem 5: $C(n,w,k) = \Theta(n + (n-w)\log k)$.

5. Computing the Smallest-Area Convex Quadrilateral

As a second application of order-statistic structures, we show that, given a set S of n points in the plane, the smallest-area convex quadrilateral whose vertices are points in S can be found in $O(n^3)$ time. This problem is closely related to an open problem of Edelsbrunner ([EDEL87], Problem 12.10): Given a set S of n points in the plane, find in $o(n^4)$ time a subset X of size four such that the area of the convex hull of X, $conv(X)$, is minimal. We show how to solve both problems in $O(n^3)$ time; a recent paper by Overmars, Rote, and Woeginger [OVER89] uses a different strategy to achieve the same time bounds.

Theorem 6: Given a set S of n points in the plane, the smallest-area convex quadrilateral can be computed in $O(n^3)$ time.

To prove this result, we consider each possible diagonal d and compute the smallest-area convex quadrilateral containing d as a diagonal.

If a and b are points, let ab be the convex hull of a and b (a line segment), and let $aff(ab)$ be the affine hull of a and b. Suppose without loss of generality that $aff(ab)$ is not vertical. Let $aff^+(ab)$ be the open region above $aff(ab)$, and let $aff^-(ab)$ be the open region below $aff(ab)$. Let $x \in S$ be a point, and consider the two cones $C_{ab}(x)$ and $C_{\overline{ab}}(x)$ with apex x and supporting lines $aff(xa)$ and $aff(xb)$. Cone $C_{ab}(x)$ contains ab while cone $C_{\overline{ab}}(x)$ does not. If a, b, and c are points, let $\Delta(abc)$ be the triangle with vertices a, b, and c.

Consider a pair of points a and b. Line segment ab may be a diagonal in a smallest-area convex quadrilateral; our algorithm removes certain points that cannot be involved in a minimum-area convex quadrilateral with ab as diameter. A point x *dominates* point y with respect to ab if x and y are on the same side of $aff(ab)$ and $y \in C_{\overline{ab}}(x)$. A *maximal point with respect to ab* is a point x that no point in S dominates; that is, no point of S is in the interior of Δabx. It is immediately apparent that if ab is a diameter of a smallest-area convex quadrilateral, we need only consider maximal points with respect to ab for the remaining two endpoints.

There is a useful property of maximal points with respect to ab — the radial order of the sequence of these points about a is the same as the radial order about b. Let $m^+(ab)$ be the set of points in $S \cap aff^+(ab)$ which are maximal. (Similar definitions and results hold for points in $aff^-(ab)$.)

Lemma 3: The clockwise radial order of $m^+(ab)$ about a is the same as the clockwise radial order of $m^+(ab)$ about b.

Proof: Suppose the order is not the same, so there are two points x and y in $m^+(ab)$ that are inverted in the two radial orderings. Then x and y must be positioned so that y dominates x. □

With Lemma 3, it is easy to state our algorithm. For each point x in S, we initially sort $S - x$ into clockwise radial order about x. We then determine $m^+(ab)$ and $m^-(ab)$, and process $m^-(ab)$ in clockwise radial order. (To determine $m^+(ab)$ we can just scan around a, starting at b, including only those pints which do not violate Lemma 3.) Let x be the first point in $m^-(ab)$ to be processed. We insert the points in $m^+(ab) \cap C_{ab}(x)$ into a min order-statistic queue Q; by Lemma 3, this set is an interval in the ordered $m^+(ab)$. Here, the key of x in Q, $key(x)$, is the perpendicular distance from x to $aff(ab)$. The point z in $m^+(ab) \cap C_{ab}(x)$ closest to $aff(ab)$ is returned by *findmin*(Q), and z is associated with x. When we consider the next point y in $m^-(ab)$, Lemma 3 implies that $m^+(ab) \cap C_{ab}(y)$ is found queue-order injections and pops on $m^+(ab) \cap C_{ab}(x)$; the point w in $m^+(ab) \cap C_{ab}(y)$ closest to $aff(ab)$ is returned by *findmin*(Q) and associated with y. This process continues until all points in $m^-(ab)$

are examined. The smallest-area convex quadrilateral with *ab* as diameter is found by choosing the pair of associated points which minimizes the sum of the perpendicular distance to *aff*(*ab*).

There are $O(n^2)$ diagonals to consider, and there are $O(n)$ inject, pop, and findmin operations for each possible diagonal *ab*, so $O(n^3)$ time is used aside from sorting and finding the maximal points. Sorting can be done in $O(n^2 \log n)$ time ($O(n^2)$ time using arrangements). Given the radial sorts, the maximal points can be found in $O(n^3)$ time, $O(n)$ time per diagonal. The details are straightforwards and are omitted. This proves Theorem 6.

There is a simple corollary to Theorem 6 which addresses the problem raised by Edelsbrunner:

Corollary 1: Given a set S of n points in the plane, a set of four points X minimizing the area of $conv(X)$ can be found in $O(n^3)$ time.

Proof: The polygon $conv(X)$ is either a triangle or a quadrilateral. If it is a quadrilateral; it can be found by the procedure used to prove Theorem 6. If $conv(X)$ is a triangle, it must contain exactly one point. Since a point x is maximal with respect to *ab* if and only if $\Delta(xab)$ is empty, the areas of nonempty triangles can be found by computing the areas of $\Delta(zab)$, where z is not maximal. Note that if a triangle contains two or more interior points, it contains a candidate for $conv(X)$ with smaller area, so no triangle containing two or more interior points can be returned as $conv(X)$, where $conv(X)$ minimizes area. □

6. Remarks

A related problem to computing running k order statistics is that of computing batched order-statistics. Given a list L of n elements, a collection C of m intervals (i_l, j_l), and integers $k_l + 1$, compute the k_l^{th} order-statistic in $L[i_l, j_l]$, $1 \le l \le m$. An efficient algorithm for computing batched order-statistics appears in Dobkin and Munro [DOBK85]. They show that L can be preprocessed in $O(n \log n)$ time and space to support $O(\log^3 n)$ time "interval" queries. An interval query consists of an interval (i, j) and an integer k, $1 \le k \le j - i + 1$, and it returns the k^{th} smallest item in $L[i:j]$. This implies that the batched order-statistic problem can be solved in $O(n \log n + m \log^3 n)$ time. We note that using the selection technique of Frederickson and Johnson [FRED84], a query can actually be made in $O(\log n + p \log k/p)$, where $p = \min\{\log n, k\}$, yielding a worst-case time of $O(n \log n + m \log^2 n)$.

We also note that our algorithm can be modified to give a more efficient algorithm for batched order-statistics when m is small and k_l is constant. To do this, we can decompose the collection of m intervals into $\sqrt{m}$ groups of intervals such that the endpoints of each group either are monotonically nondecreasing or "concentrically contained." (This relies on the theorem that any poset of m elements either contains a chain of size $\Omega(\sqrt{m})$ or an antichain of size $\Omega(\sqrt{m})$ [LIU85].) It takes $O(m^{3/2})$ time to decompose C, and $O(n \log k)$ time for each decomposed problem, for an $O(m^{3/2} + \sqrt{m}\, n \log k)$ time algorithm.

7. References

[ATKI86] M. D. Atkinson, J. Sack, N. Santoro and T. Strothotte, Min-Max Heaps and Generalized Priority Queues, *Comm. ACM*, 29(10), 1986, pp. 996-1000.

[DOBK85] D. P. Dobkin and J. I. Munro, Efficient Uses of the Past, *Journal of Algorithms*, **6**, 1985, pp. 455-465.

[EDEL87] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer Verlag, 1987.

[FRED84] G. N. Frederickson and D. B. Johnson, Generalized Selection and Ranking: Sorted Matrices, *Siam J. on Computing*, 13(1), 1984, pp. 14-30.

[FRED87] M. L. Fredman and T. H. Spencer, Refined Complexity Analysis for Heap Operations, *J. Computer and System Sciences*, **35**, 1987, pp. 269-284.

[GAJE86] H. Gajewska and R. E. Tarjan, Deques with Heap Order, *Inf. Proc. Letters*, **22**, 1986, pp. 197-200.

[HARE84] D. Harel and R. E. Tarjan, Fast Algorithms for Finding Nearest Common Ancestors, *Siam J. on Computing*, 13(2), 1984, pp. 329-337.

[LIU85] C. L. Liu, *Elements of Discrete Mathematics*, McGraw-Hill, 1985.

[MARA87] P. Maragos and R. W. Schafer, Morphological filters — Part II: Their relations to median, order-statistic, and stack filters, *IEEE Trans. Acoust., Speech, Signal Processing*, **ASSP-35**, 1987, pp. 1170-1184.

[OVER89] M. Overmars, G. Rote and G. Woeginger, Finding Minimum Area *k*-gons, *Free University of Berlin, B-89-02*, 1989.

[RICH90] D. S. Richards, VLSI Median Filters, *IEEE Trans. Acoust., Speech, Signal Processing*, **ASSP-38**, 1990, pp. 145-153.

[SHAM76] M. I. Shamos, Geometry and Statistics: Problems at the Interface, in *Algorithms and Complexity: New Directions and Recent Results*, J. F. Traub (editor), Academic Press, New York, NY, 1976, 251-280.

[VUIL80] J. Vuillemin, A Unifying Look at Data Structures, *Comm. ACM*, 23(4), 1980, pp. 229-239.

[WILL64] J. W. J. Williams, Algorithm 232, *Comm. ACM*, 7(6), 1964, pp. 347-348.

ON SOME COMBINATORIAL OPTIMIZATION PROBLEMS:
Bottleneck and Capacity Assignment Problems

Wojciech Szpankowski*
Department of Computer Science
Purdue University
W. Lafayette, IN 47907
U.S.A.

Abstract

The bottleneck (capacity) assignment problem minimizes (maximizes) the largest (smallest) element over all permutations in a matrix A. Our interest lies in finding the asymptotically exact solution to these problems in a probabilistic framework, that is, under the assumption that elements in a matrix are independent random variables with common continuous distribution function $F(\cdot)$. It is proved that the optimal values for the bottleneck and capacity assignment problems asymptotically become $F^{-1}(\log n/n)$ and $F^{-1}(1 - \log n/n)$ *in probability* respectively, where n is the size of the matrix. Finally, using the above probabilistic results we design a simple heuristic algorithm that is asymptotically optimal, however, the cost of the heuristic is much cheaper than the optimal algorithm.

1. INTRODUCTION

Most algorithms are designed to optimize the worst-case performance. Many elegant constructions have been set up in this endeavor. We note, however, that such a construction has to cope efficiently with unrealistic, even pathological inputs and the possibility is neglected that a simpler algorithm might perform just as well, or even better in practice. One possible solution is to look at an algorithm from the probabilistic view point, so rather typical inputs instead of pathological ones are investigated. This probabilistic approach to design algorithms was practically fulfilled a decade ago when it became clear that the prospects for showing the existence of polynomial algorithms for NP-hard problems were very dim. This fact and apparently a high success rate of heuristic approaches applied not necessary to NP-hard problems, led computer scientists to undertake a more serious investigation of probabilistic approximation algorithms [KA].

In this paper we investigate the capacity and bottleneck assignment problems (we further refer to them as CAP and BAP respectively) in a probabilistic framework [GG,AV,HS]. These problems can be formulated as follows. Let $A = \{a_{ij}\}_{i,j=1}^{n}$ be a $n \times n$ matrix of real numbers which we further call *weights*. In the bottleneck (capacity) assignment problem, we ask to minimize (maximize) the largest (smallest) element over all possible sets of n entries in A, one

---

*This research was supported by AFOSR Grant 90-0107, and in part by the NSF Grant CCR-8900305, and by Grant R01 LM05118 from the National Library of Medicine.

from each row and column (i.e., a permutation). In our probabilistic framework we assume that the elements of the matrix A are selected randomly and independently with distribution function $F(\cdot)$ that is (strictly) continuous. Under this assumption, we shall show that the optimal values of the BAP and CAP converge *in probability* and *in mean* to $F^{-1}(\log n/n)$ and $F^{-1}(1 - \log n/n)$ respectively. This settles the problems left open in [GG] and [WE]. As a consequence of our solution, we construct a heuristic algorithm which is asymptotically optimal with high probability, that is, the relative error between the heuristic and the optimal solutions tends to zero in some probability sense as n becomes large. However, our heristic algorithm runs in $O(n^2)$ steps instead of $O(n^3)$ steps required for the optimal solution (see some improvement in [GT]).

The bottleneck assignment problem (BAP) in a probabilistic framework was discussed in [GG,AV,HS,GT], and also shortly in [LU,WE]. In particular, Garfinkel and Gilbert [GG] have proved a lower bound for uniformly distributed weights that matches our lower bound. Weide [WE] has shown also an upper bound of the form $F^{-1}(C \log n/n)$, however, the constant C was not determined. Actually, for Weide to prove his result he had to use a powerful result of Pósa from a random graph theory [BO]. Our approach is completely different and we use only elementary property of the problem, and some results from order statistics [GA]. Nevertheless, the methodological approach adopted in this paper is *not* restricted to the assignment problems. In particular, it can be applied to the analysis of traveling salesman problems, spanning trees, geometric location problems, minimum weighted clique problems and so on [SZ1, SZ2].

The paper is organized as follows. In the next section, we present our main results and discuss some consequences of them. Proofs are delayed till Section 3. In addition, the last section contains a description of a heuristic algorithm to solve the problem, as well as some computer experiments that confirm our theoretical results.

2. MAIN RESULTS

Let $A = \{a_{ij}\}_{i,j=1}^{n}$ be an $n \times n$ matrix of real numbers (weights) and by $\sigma(\cdot)$ we denote a permutation of the set of indices $\{1, 2, ..., n\}$. The set of all permutations of $\{1, 2, ..., n\}$ is denoted by B_n, and naturally the cardinality of B_n is $n!$, that is, $|B_n| = n!$ The bottleneck assignment problem (BAP) seeks such a permutation σ that minimizes $\max_{1 \le i \le n} a_{i,\sigma(i)}$. That is, the optimal value of the objective function Z_{min} for (BAP) is

$$Z_{min} = \min_{\sigma \in B_n} \max_{1 \le i \le n} a_{i,\sigma(i)} \tag{2.1}$$

On the other hand, the optimal value of the objective function Z_{max} for the capacity assignment problem (CAP) is a reverse to (2.1), that is,

$$Z_{max} = \max_{\sigma \in B_n} \min_{1 \le i \le n} a_{i,\sigma(i)} \tag{2.2}$$

The formulation (2.1) and (2.2) are in fact more general than needed for BAP and CAP problems. For example (2.1) can be viewed as the bottleneck traveling salesman problem, if one interprets B_n as as a set of all Hamiltonian circuits in a graph span over n vertices and a_{ij} is understood as a weight assigned to the (i,j)-edge. In the same spirit (2.2) is a correct formulation for the capacity traveling salesman problem [SZ1, SZ2]. In fact, interpreting the weights a_{ij} in the last examples as the capacity of the edge (i,j), we can nicely motivate the names given to these two problems. More importantly, our main results hold also for the bottleneck and capacity traveling salesman problems.

We shall analyze both problems in a probabilistic framework by assuming that the weights a_{ij} are *independently* and *identically* distributed (i.i.d.) random variables with common distribution functions $F(\cdot)$. In addition, we assume that $F(x)$ is an *increasing* function of x. (Note that $F(\cdot)$ is always nondecreasing as a distribution function, but in addition, we require that $F(\cdot)$ is strictly increasing which implies that $F(\cdot)$ is strictly continuous.) By the last assumption, we note that

$$F(Z_{min}) = \min_{\sigma \in B_n} \max_{1 \le i \le n} F(a_{i,\sigma(i)}) \tag{2.1a}$$

and the same holds for (2.2). This reflects the fact that the solution of (2.1) and (2.2) does not depend on the individual values of a_{ij}, but only the ranking of the solution is important. This also implies that proving optimality of the problem for one particular distribution is enough to obtain a general solution. In particular, we shall use the fact that for any random variable X with the distribution $F(\cdot)$ the mapping $F(X) = U$ transforms X into a uniformly distributed random variable U [RE]. It will be convenient, as we shall see later, to work at the beginning with the exponential distribution.

Our main results can be formulated as follows.

THEOREM *(i) For the bottleneck assignment problems, the solution Z_{min} in (2.1) converges in probability to the following constant $F^{-1}(\log\ n/n)$, where* log *represents the natural logarithm. More precisely, as n tends to infinity*

$$\lim_{n\to\infty} \frac{Z_{min}}{F^{-1}(\log\ n/n)} = 1 \qquad \textit{in probability} \tag{2.2}$$

In addition, if the r-th moment of the weights a_{ij} exists, then for large n

$$EZ^r_{min} = (1+o(1))[F^{-1}(\log\ n/n)]^r \tag{2.3}$$

where EZ^r_{min} stands for the r-th moment of Z^r_{min}.

(ii) For the capacity assignment problem, the following holds for large n

$$\lim_{n\to\infty} \frac{Z_{max}}{F^{-1}(1-\log\ n/n)} = 1 \qquad \textit{in probability} \tag{2.4}$$

In addition, assuming that the r-th moment of the weights exists, then

$$EZ^r_{max} = (1 + o(1))[F^{-1}(1-\log\ n/n)]^r \tag{2.5}$$

where EZ^r_{max} is the r-th moment of Z_{max}. ∎

We delay the proof of the theorem and all necessary technicalities to Section 3. The rest of this section is devoted to provide an interpretation of our theorem, and to discuss some algorithmic consequences. More details can be found at the end of Section 3.

As far as interpretation is concerned, condition (2.2) implies that the following probability $\Pr\{|Z_{min}/F^{-1}(\log n/n) - 1| > \epsilon\}$ becomes smaller and smaller for larger values of n, that is, $\lim_{n\to\infty} \Pr\{|Z_{min}/F^{-1}(\log n/n) - 1| > \epsilon\} = 0$. Roughly speaking, this means that it is very unlikely that Z_{min} differs from $F^{-1}(\log n/n)$ by more than ϵ, whatever $\epsilon > 0$ is selected. On the other hand, condition (2.3) implies that $\lim_{n\to\infty} EZ^r_{min}/[F^{-1}(\log n/n)]^r = 1$, so the r-th moment of the objective function Z_{min} is well approximated by $[F^{-1}(\log n/n)]^r$ for large n. These facts imply that for large n the objective function (a random variable) Z_{min} becomes a degenerate random variable such that $Z_{min} \sim EZ_{min}$ *in probability.* This suggests also that any algorithm that produces a solution within an order of magnitude of $F^{-1}(\log n/n)$ is acceptable in the sense that the probability of error between this algorithm and the optimal solution becomes zero for large n (for more details see error lemma in [WE]). We next concentrate on this observation, and present an overview of our algorithmic consequences, while more details the reader can find in Section 3.

A brute force solution to either BAP or CAP requires $O(n!)$ time complexity, however, a clever solution of Edmonds [ED] and Gabow [GB] needs only $O(n^3)$ (see [GT] for some improvements). A question arises whether a heuristic algorithm can produce *faster* a solution which *is close* to the optimal one. In fact, such a heuristic is easy to construct! Indeed, at first we sort all n^2 elements of A in an increasing order. Then, one finds such $m^* < n^2$ that *with probability one* there exists a feasible solution (a permutation) in a non-full matrix built from m^* elements (we shall prove that $m^* = n \cdot \log n + cn$ suffices to satisfy this requirement). Such an heuristic is adopted as an approximation for the optimal algorithm since we can easily prove that this heuristic produces a solution that is asymptotically equal to $F^{-1}(\log n/n)$ (see Section 3), that is, asymptotically the same as the optimal solution! But, our heuristic running time can be reduced to $O(n^2)$ by using a heap to generate increasing order of weights until m^* is reached, and using Avis and Lai heuristic algorithm [AL] for checking feasibility of a solution (i.e., finding a matching in an equivalent bipartite graph).

3. ANALYSIS AND ALGORITHMIC CONSEQUENCES

In this section we prove our theorem. We shall concentrate on BAP problems, and during the analysis we propose a methodology which can be easily extended to the CAP problem.

To prove our estimate on Z_{min} defined in (2.1), we bound the objective function from the below and from the above by two other random variables which are easier to evaluate. For the lower bound $\underline{Z}$ we select the minimum value in each column of A, and then take the maximum over all such elements. That is,

$$\underline{Z} = \max_{1 \le j \le n} \min_{1 \le i \le n} a_{ij} \le \min_{\sigma \in B_n} \max_{1 \le i \le n} a_{i,\sigma(i)} = Z_{min} \tag{3.1}$$

The upper bound for Z_{min} is conceptually easy to construct. Indeed, selecting a particular permutation, say σ^*, by the definition of the problem we must have $Z_{min} \le Z(\sigma^*)$ where $Z(\sigma^*)$ is the value of the objective function $Z(\cdot)$ for the permutation σ^*. As a particular solution to our problem we use the following algorithm that is further referred as BOTTLE, and the algorithm is of its own interest (cf. [HS]).

```
Algorithm BOTTLE

begin
    sort weights such that $a_{(1)} \le a_{(2)} \le ... \le a_{(n^2)}$
    $i = 0$
    repeat
            begin
            $i = i + 1$
            add $a_{(i)}$ element to the structure built so far
            build a partial permutation $\beta$ (not necessary a permutation $\sigma$)
            end
    until $\beta$ becomes a permutation $\sigma$
output $\sigma$
end
```

Let Z_{bottle} denote the value of the objective function for the above algorithm. Naturally,

$$Z_{bottle} = \max_{1 \le j \le m^*} a_j \ge Z_{min} \tag{3.2}$$

where $m^* < n^2$ represents the minimum number of elements of A – selected randomly from A - that assures the existence of a permutation $\sigma \in B_n$. Then, finally

$$\underline{Z} \le Z_{min} \le Z_{bottle} \tag{3.3}$$

and the problem is solved once we establish some asymptotics for order statistics defined in (3.1) and (3.2), namely the maximum of independent, however not necessary identical distributed, random variables. We address below this problem in a more general setting by establishing in the Lemma below a simple tool to deal with maximum of (even dependent) random variables.

Let X_1, $X_2, ..., X_n$ be a sequence of random variables each distributed according to $G_1(\cdot)$,..., $G_n(\cdot)$. Define

$$M_n = \max_{1 \le i \le n} X_i \tag{3.4}$$

We are interested in the behavior of M_n for large values of n. In particular, we shall study convergence of M_n *in probability* sense. We restrict the class of distribution functions $G_1(x)$, $G_2(x), ..., G_n(x)$ to ones satisfying the following two conditions

$$\begin{aligned} &\text{(i)} \quad \text{for every } k = 1, 2, ..., n \quad G_k(x) < 1 \quad \text{for} \quad x < \infty \\ &\text{(ii)} \quad \text{uniformly in } k = 1, 2, ..., n \text{ the following holds} \end{aligned} \tag{3.5}$$

$$\lim_{x \to \infty} \frac{1 - G_k(cx)}{1 - G_k(x)} = 0 \quad \text{for} \quad c > 1 \tag{3.6}$$

Let us also define a_n as the smallest solution to the next equation

$$\sum_{k=1}^{n} [1 - G_k(a_n)] = 1 \tag{3.7}$$

We prove the following lemma which is a slight extension of the results presented in [GA].

Lemma *(i) If conditions (3.5) and (3.6) hold, then*

$$\lim_{n\to\infty} \frac{M_n}{a_n} \le 1 \quad in\ probability \tag{3.8}$$

where a_n is the solution of (3.7).

(ii) If, in addition to (3.5) and (3.6), $X_1, X_2, ..., X_n$ are independent, then

$$\lim_{n\to\infty} \frac{M_n}{a_n} = 1 \ in\ probability \tag{3.9}$$

(iii) Assume hypotheses of (ii) hold and the r-th moments of $X_1, X_2, ..., X_n$ exist, then

$$EM_n^r = (1 + o(1))a_n^r \tag{3.10}$$

and a_n is defined in (3.7).

Proof. We first prove part (i). Let $R_k(x) = 1 - G_k(x) = \Pr\{X_k < x\}$. Note that, taking also into account (3.5), we have [RE]

$$\Pr\{M_n > r\} = \Pr\{X_1 > r \text{ or } X_2 > r \text{ or } , ..., \text{ or} X_n > r\} \le \sum_{k=1}^{n} R_k(r) \ . \tag{3.11}$$

Put $r = a_n(1 + \epsilon)$, and note that (3.6) implies $R_k(a_n(1 + \epsilon)) = o(1)R_k(a_n)$ uniformly in k, so from (3.11) we obtain

$$\Pr\{M_n > (1+\epsilon)a_n\} = o(1)\sum_{k=1}^{n} R_k(a_n) = o(1)$$

by (3.7). This proves Lemma (i).

To prove (3.9) we assume, in addition that X_1, $X_2, ..., X_n$ are independent, that is,

$$\Pr\{M_n < r\} = \Pr\{X_1 \le r, X_2 \le r , ..., X_n \le r\} = G_1(r) \ ... \ G_n(r) \tag{3.12}$$

It is more convenient to deal with logarithm of (3.12). We first note that for $v \to 1$ [GA] $\log v = \log[1 - (1 - v)] \le v - 1$, hence for (3.12)

$$\log \Pr\{M_n < r\} \le -\sum_{k=1}^{\infty} [1 - G_k(r)] \tag{3.13}$$

Let $r = (1 - \epsilon)a_n$, $\epsilon > 0$. Then to prove (3.9), we need to show that the RHS of (3.13) tends to $-\infty$ when $n \to \infty$ for $r = (1 - \epsilon)a_n$. But substituting in (3.6) $x = z/c$ for $c > 1$ one finds $1 - G_k(x) = o(1)\ (1 - G_k(z/c))$. Let $1/c = 1 - \epsilon < 1$, then by (3.7) and the above one shows

$$\sum_{k=1}^{n} [1 - G_k(1-\epsilon)a_n)] = \frac{1}{o(1)} \sum_{k=1}^{n} [1 - G_k(a_n)] = \frac{1}{o(1)} \tag{3.14}$$

so this proves $\log \Pr\{M_n < (1 - \epsilon)a_n\} \to -\infty$, and it completes the derivation of (3.9).

In Lemma (ii), we have proved that $M_n \to a_n$ (pr). This would imply convergence *in mean* (3.10) if one proves that X_1, $X_2, ..., X_n$ are uniformly integratable [RE]. This fact is proved in [GA] assuming (3.6) holds and X_1, $X_2, ..., X_n$ possess r-th moments. ■

Now we are ready to prove our main result, that is, Theorem from Section 2. We first consider the lower bound $\underline{Z}$ given by (3.1). Let $X_k = \min_{1 \le i \le n} a_{ij}$ where a_{ij} is $F(\cdot)$ distributed. Note that $G_k(x) = \Pr\{X_k < x\} = [1 - F(x)]^n$, since a_{ij} are i.i.d. To apply our lemma, we must select a distribution $F(\cdot)$ satisfying (3.5) and (3.6). Since, as noticed above in (2.1a), the solution to the BAP problem is distribution-independent, we can select any strictly continuous distribution without loss of generality. Therefore, we assume a_{ij} are exponentially distributed, that is $F(x) = 1 - e^{-x}$. The last step before applying our lemma, is to solve equation (3.7). In our case it becomes

$$ne^{-na_n} = 1$$

We note that $U = F(X)$ is uniformly distributed, so for $F(x) = 1 - e^{-x}$ we find that a_n for uniformly distributed weights is

$$a_n = 1 - \exp(-\log\ n/n) = \frac{\log\ n}{n} + o(\log n/n) \tag{3.15}$$

This directly implies that $Z_{min} \ge F^{-1}(\log n/n + o(\log n/n))$ for any continuous distribution function $F(\cdot)$ [RE].

To prove the upper bound, we proceed as follows. Let $a_{(1)} \le a_{(2)} \le ... \le a_{(n^2)}$ denote all n^2 weights ordered in an increasing sequence. Then, according to the algorithm BOTTLE the optimal solution to a bottleneck optimization problem is found after inspecting m^* elements (cf. (3.2)). Naturally, $Z_{\min} \le w_{(m^*)}$. In our probabilistic framework, we assume that the weights W_i (capital letters denote random variables) for $1 \le i \le n^2$ are uniformly distributed i.i.d. (independently, identically distributed) random variables, and we denote by M_n^* the number of elements necessary to inspect in order to construct *almost surely* a permutation. Note that the M_n^*-th order statistic of, say, n^2 uniformly distributed random variables W_i is $W_{(M_n^*)} = M_n^*/(n^2 + 1) + o(1)$ (pr.), that is, for any ϵ the following holds $\lim_{n\to\infty} \Pr\{|W_{M_n^*} - M_n^*/(n^2 + 1)| > \epsilon\} = 0$. This is a simple consequence of the following two elementary facts [GA]:

- The r-th order statistic $U_{(r)}$ of m uniformly distributed i.i.d. random variables satisfies

  $$EU_{(r)} = \frac{r}{m+1} \quad ; \quad varU_{(r)} = \frac{r(m-r+1)}{(m+1)^2(m+2)}\ , \tag{3.16}$$

  hence by Chebyshev's inequality $U_{(r)} \to EU_{(r)}$ (pr.) as $m \to \infty$. In fact, by Borel-Cantelli lemma the convergence *in probability* may be replaced by *almost sure* convergence provided $r = o(m)$

- The above property holds even if r is replaced by a random variable R_m such that $R_m/r \to 1$ (a.s.)

Then, our claim follows from the Chebyshev's inequality, and therefore $Z_{\min} \le W_{(M_n^*)}$ (pr.). In summary, we just proved that as $n \to \infty$

$$Z_{\min} \le \frac{M_n^*}{n^2} \quad (pr.)\ , \tag{3.17a}$$

for uniformly distributed weights W_i. This and (3.1) further imply that for any other distribution function $F(\cdot)$

$$Z_{\min} \leq F^{-1}(M_n^*/n^2) \quad (pr.) \tag{3.17b}$$

provided $F^{-1}(\cdot)$ is strictly continuous (cf. [RE]). To complete the prove we need a formula on M_n^*. This can be easily found by reducing the problem to an urn-and-balls problem. Indeed, we note that M_n^* is the minimum number of balls thrown to n urns that assures all urns contain at least one ball. But, Holst [HO] proved that and $M_n^* = n(\log n + \omega_n)$ where $\omega_n \to \infty$. Hence, (3.7b) and the above completes the proof of our main Theorem.

Our main result, in fact, suggest a simple heuristic algorithm for BAP, as observed before. This algorithm is a modification of BOTTLE that avoids the expensive **repeat-until** loop. The idea of the heuristic is presented below, where we use the fact that a permutation in a matrix correspond to a perfect matching in an equivalent bipartite graph with weights A.

```
Algorithm HEURISTIC

begin
    sort weights such that $a_{(1)} \leq a_{(2)} \leq ... \leq a_{(n(n-1)/2)}$
    Set $m = m^*$
    find a perfect matching in the equivalent bipartite graph $G_{n,m^*}$
output $\alpha$ or NOT FOUND
end
```

In the above m^* should assure with probability one the existence of a permutation $\sigma \in B_n$, that is, $m^* = n(\log n + \omega_n)$ [HO]. In practice, instead of $m = m^*$ we run the algorithm for a couple of iterations from $m = m^* - O(1)$ to $m = m^* + O(1)$. Naturally, the time complexity of the algorithm heiristic is $O(n^2)$.

In order to visualize and verify our theoretical results we have programmed our algorithms BOTTLE and HEURISTIC for the BAP problem. In BOTTLE we used an improved *Hungarian Method* to check whether a perfect matching exists or not. In both algorithms BOTTLE and HEURISTIC we build heap to sort efficiently (in $O(n^2)$ steps) weights w_{ij}. Finally, we implemented in HEURISTIC a simple and effective (time-complexity of $O(n^2)$) subalgorithm to inspect whether the selected weights cover the whole matrix A, that is, whether there is at least one weight in every column and every row (note that this assure that m^* is properly selected).

We have used three different distributions for weights, namely normal distribution $N(0,1)$, gamma distribution $gamma(\lambda,\beta)$, and beta distribution $beta(\alpha_1,\alpha_2)$. For each distribution we evaluated the optimal value using our exact algorithm BOTTLE and compare it with the theoretical optimal value obtained from our Theorem.

From the table one may immediately note very good accuracy of our theoretical results even for small size of the problem ($100 \leq n \leq 500$). This implies good convergence rates for our theoretical results. In additional, this suggests a good quality of our heuristic algorithms, and it

Table 1: Comparison of simulation and theoretical results.

DISTRIBUTION	SIZE	VALUE			TIME	
		THEORY	OPTIMAL	HEURISTIC	BOTTLE	HEURISTIC
Normal (0,1)	100	-1.500000	-1.62355	-1.48766	77	8
	200	-1.770000	-1.860940	-1.87524	736	58
	300	-1.920000	-1.957682	-1.99665	3285	117
	400	-2.020000	-2.098472	-2.05880	7758	288
	500	-2.100000	-2.161564	-2.16474	15002	566
Exponential (1)	100	0.047146	0.053723	0.05328	79	34
	200	0.026849	0.033738	0.02819	741	43
	300	0.019196	0.026014	0.02601	3527	135
	400	0.015092	0.016736	0.01559	6495	298
	500	0.012507	0.015816	0.01582	18070	411
Uniform (0,1)	100	0.046052	0.043567	0.04347	69	7
	200	0.026492	0.032597	0.03373	869	41
	300	0.019013	0.026200	0.02593	3456	139
	400	0.014979	0.018658	0.01866	7340	303
	500	0.012429	0.015084	0.01405	16085	481

was confirmed by computer runs. From the table we note that the running time for BOTTLE is approximately $121n^3$, while for HUERISTIC is only $15n^2$ which is significant time saving even for moderate values of n. Furthermore, we point out that our algorithm BOTTLE does not differ significantly from an optimal algorithm for this problem, and our another computer experiments confirm this observation (hence our heuristic is even more valuable).

The proof of the second part of our Theorem is derived in a similar way. This time, however, we must concentrate on minimum of a set $X_1, X_2, ..., X_n$ of random variables. A lemma similar to the one proved in the paper can be formulated for $m_n = \min\{X_1, ..., X_n\}$. We need, however, to replace in (3.5) - (3.7) $1 - G_k(x)$ by the distribution function $G_k(x)$. In particular, (3.7) becomes

$$\sum_{k=1}^{n} G_k(a_n) = 1 \tag{3.18}$$

The other steps of the previous proof are the same. In particular, a solution to (3.18) for the negative exponential distribution becomes $a_n \sim 1 - \log n/n$, as needed in Theorem (ii). The upper bound is proved as before by using the algorithm BOTTLE with weights ordered in an non-increasing sequence.

REFERENCES

[AL] D. Avis and C.W. Lai, The Probabilistic Analysis of a Heuristic for the Assignment Problem, *SIAM J. Computing,* 17, 732–741, 1988.

[AV] D. Angluin and L. Valiant, Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings, *J. Computer and System Sciences,* 18, 155–193, 1979.

[BO] B. Bollobas, *Random Graphs,* Academic Press, London 1985.

[ED] J. Edmonds, Paths, Trees and Flowers, *Canad. J. Math.,* 17, 449 - 467, 1965.

[GA] J. Galambos, *The Asymptotic Theory of Extreme Order Statistics,* R.E. Krieger Publication Company, Malabar, 1987.

[GB] H. Gabow, An Efficient Implementation of Edmonds' Algorithm for Maximum Matching on Graphs, *Journal of the ACM,* 23, 221-234, 1976.

[GT] H. Gabow and R. Tarjan, Algorithms for Two Bottleneck Optimization Problems, *J. Algorithms,* 411-417, 1988.

[GG] R. Garfinkel and K. Gilbert, The Bottleneck Traveling Salesman Problem: Algorithms and Probabilistic Analysis, *J. ACM,* 25, 435–448, 1978.

[HO] L. Holst, On Birthday, Collector's, Occupancy and other Classical Urn Problems, *International Statistical Review,* 54, 15-27, 1986.

[HS] D. Hochbaum and D. Shmoys, A Unified Approach to Approximation Algorithms for Bottleneck Problems, *J. of the ACM,* 33, 533–550, 1986.

[KA] Karp, R., The Probabilistic Analysis of Some Combinatorial Search Algorithms, in: *Algorithms and Complexity,* (ed., J.F. Traub), Academic Press, New York, 1976.

[LU] Lueker, G., Optimization Problems on Graphs With Independent Random Edge Weights, *SIAM J. Computing,* 10, pp. 338–351, 1981.

[RE] Rényi, A., *Probability Theory,* North-Holland, Amsterdam, 1970.

[SZ1] Szpankowski, Z., (Probability) Optimal Solution to Some Problems Not Only on Graphs, Purdue University CSD-TR-780, 1988; revised CSD-TR-872, 1989.

[SZ2] Szpankowski, Z., Another Unified Approach to Some Bottleneck and Capacity Optimization Problems, Purdue University CSD-TR-1022, 1990.

[WE] Weide, B., Random Graphs and Graph Optimization Problems, *SIAM J. Computing,* 9, pp. 552–557, 1980.

Minimizing the maximum wire length in VLSI graph layouts

Deepak D. Sherlekar
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122

Abstract: The layout area of a graph depends primarily on the number of edges cut at each step to recursively split it into partitions of nearly equal size. This property is influenced most by parameters such as the size of the nodes, and if the graph is planar, by its resemblance to an outerplanar graph. The latter is measured by the *gauge* of the graph. We have previously developed area-efficient layout schemes which are sensitive these parameters. In this paper, we extend these schemes to minimize the maximum wire length in these layouts. We consider arbitrary graphs possessing good separators and bifurcators, as well as planar graphs. Our layouts are optimal for x^α separable graphs for $\alpha < 1/2$ and $\alpha > 1/2$, and are nearly optimal in other cases. For planar graphs, our results span the spectrum between outerplanar graphs, which have gauge 1, and arbitrary planar graphs, which may have gauge $\Omega(N)$, while simultaneously allowing vertices of arbitrary degree. The optimality of our method is examined by proving lower bounds on the maximum wire length of such graphs. In cases where we cannot establish optimality, our bounds place previous results in context by demonstrating gaps between the lower and upper bounds which are sensitive to these parameters. Moreover, we can establish stronger lower bounds in these cases for corresponding nonplanar graphs having identical partitioning characteristics.

1 Introduction

The classical model for VLSI layouts [10] maps vertices of a circuit graph to points on the integer grid, and edges to edge disjoint paths between these points. Under this model, vertices occupy unit area, edges have unit thickness, and no vertex has degree greater than 4. However, these assumptions may be violated in hierarchical design methods such as macro-cell based layout schemes, where vertices correspond to modules. We have developed area-efficient layouts which reflect these considerations in [9] and [8]. In this model, larger vertices are represented by regions of appropriate area, and a vertex of degree *d* by a square region of perimeter at least *d*. The objective of this paper is to develop methods to minimize the maximum wire length in these layouts, and to investigate the optimality of these methods through corresponding lower bounds.

Research on VLSI graph layouts [5, 1, 4, 9] shows that the layout area of a graph is affected most by its recursive partitioning characteristics and the size of its nodes. For a planar graph, an important factor influencing the former is its degree of resemblance to an outerplanar graph, as measured by its *gauge* [3, 8]. A planar graph is called outerplanar if it can be drawn in the plane so that no two edges cross each other, and all vertices lie on the cycle forming the boundary of the drawing (the outer face). A planar graph has *gauge* F if it has an embedding in the plane in which any vertex can reach a vertex on the outer face by a path of length at most $F - 1$. Thus outerplanar graphs have gauge 1. There exist planar graphs, such as the *diamond graph* [11] which have gauge $\Theta(N)$.

[5, 11] have developed $O(N \log^2 N)$ area layouts of planar graphs of bounded degree. We have presented corresponding gauge-sensitive layouts of area $O(N \log^2 F)$ in [8]. The tightest corresponding lower bounds known are $\Omega(N \log N)$ [4], and $\Omega(N \log F)$ [7] respectively. We have also developed area-efficient layouts for planar graphs of arbitrary degree and gauge in [8], and presented corresponding lower bounds in [7]. For graphs whose partitioning characteristics are captured by bifurcators, area optimal layouts for the bounded degree case appear in [1], and for the arbitrary degree case in [9].

Gauge F	Upper Bound on the maximum wire length	Gap between upper and lower bound	
		planar graphs	arbitrary graphs
$O(1)$	$O(\sqrt{N} / \log N)$ [6, 2]	$O(1)$ [6]	$O(1)$ [6]
$O(N)$	$O(\sqrt{N} \log N / \log \log N)$ [2]	$O(\log^{3/2} N / \log \log N)$ [4]	$O(1)$ [4]
PARAMETER $O(\sqrt{N} / \log N)$	$O(\sqrt{N} \log F / \log(N/F^2))$ *[Theorem 3.1]*	$O(\log N \sqrt{\log F} / \log(N/F^2))$ *[Theorem 3.2]*	$O(1)$ *[Section 3]*

Table 1: Bounds on the maximum wire length for bounded degree planar graphs.

The problem of minimizing the maximum wire length while limiting the increase in area to a constant factor has also been addressed. However, our work on gauge and degree sensitive layouts of planar graphs and graphs having bifurcators did not consider minimizing the maximum wire length. We adapt the method in [1] to extend our layout schemes to address this problem, and prove corresponding lower bounds. Our results are summarized in Tables 1 and 2 along with previous results on this problem. Note that the lower bounds for planar graphs are not tight. However, the upper bounds for planar graphs are also valid for nonplanar graphs having identical partitioning characteristics. [4] has demonstrated a nonplanar graph (the mesh of trees) which is $\sqrt{x}$-separable like planar graphs, and for which the gap between the upper and lower bounds is tight. For planar graphs of any gauge F we define a graph based on the mesh of trees which has the same partitioning characteristics, and prove a tight lower bound on its maximum wire length.

Dissection Mechanism	Maximum Degree (Δ)	Gauge(F) (if planar)	Maximum wire length	
			Upper Bound	Lower Bound
$(B, \sqrt{2})$ bifurcator	PARAMETER	—	$O\left(\frac{B \log(N/B)}{\log_\Delta \log(N/B)}\right)$ *[Theorem 4.1]*	$\Omega\left(\frac{B \log(N/B)}{\log \log(N/B)}\right)$ *[Theorem 4.1]*
x^α, $\alpha < 1/2$ Separator	PARAMETER	—	$O(\frac{\sqrt{\Delta^{1/(1-\alpha)} N}}{\log_\Delta N})$ *[Theorem 4.2]*	$\Omega(\frac{\sqrt{\Delta^{1/(1-\alpha)} N}}{\log_\Delta N})$ *[Theorem 4.2]*
Planar Separators	PARAMETER	$\Omega(\sqrt{N/\Delta}/ \log N)$	$O\left(\frac{\sqrt{\Delta N} \log(N/\Delta)}{\log_\Delta \log N}\right)$ *[Theorem 4.3]*	$\Omega\left(\frac{\sqrt{\Delta N \log(N/\Delta)}}{\log_\Delta N}\right)$ *[Theorem 4.5]*
Planar Separators	PARAMETER	$O(1)$	$O(\sqrt{\Delta N} / \log_\Delta N)$ *[Theorem 4.3]*	$\Omega(\sqrt{\Delta N} / \log_\Delta N)$ *[Theorem 4.4]*
Planar Separators	PARAMETER	PARAMETER $O(\sqrt{N/\Delta}/ \log N)$	$O\left(\frac{\sqrt{\Delta N} \log F}{\log_\Delta (N/F^2)}\right)$ *[Theorem 4.3]*	$\Omega\left(\frac{\sqrt{\Delta N \log F}}{\log_\Delta N}\right)$ *[Theorem 4.5]*

Table 2: Bounds on the maximum wire length for graphs of arbitrary degree.

Section 2 outlines our basic method. Gauge-sensitive layouts for bounded degree planar graphs are the topic of Section 3. As shown in Table 1, the gap between our lower and upper bound on the maximum wire length is a function of F and N, and spans the spectrum between the previously known results for gauge $O(1)$ and gauge $O(n)$. Moreover, as in [4], we demonstrate the existence of nonplanar graphs having identical partitioning characteristics for which the bounds achieved by our methods are optimal, up to constant factors. Nonplanar and planar graphs of arbitrary degree are considered in Section 4. These results are summarized in Table 2. Concluding remarks appear in Section 5.

2 An Outline of the Method

The basic method to minimize the maximum wire length involves a two-stage process, and is adapted from [2, 1]. The method applies to layouts found by first embedding the graph in a graph resembling the *tree of meshes*. The *tree of meshes* T_n [4] is a planar graph resembling a binary tree whose vertices have been replaced by meshes. The mesh at the root is of size $n \times n$. The number of rows in the meshes on subsequent levels are halved at every odd level, and the number of columns are halved at every even level. The tree of meshes T_4 is shown in Figure 1. T_n has $N = 2n^2 \log n + n^2$ vertices, gauge $F = n = \Theta(\sqrt{N/\log N})$, and layout area $\Theta(n^2 \log^2 n) = \Theta(N \log N)$. The layouts we developed in [9, 8] use a generalization of T_n. In this graph, the sizes of the meshes at every level i are a function of i and the graph dissection mechanism being used for the embedding. However, this does not affect the basic strategy. The first stage of the method minimizes the number of meshes spanned by an edge. The second minimizes the inter-mesh wire length, i.e., the length of the longest wire between two meshes. The two operations are described briefly below.

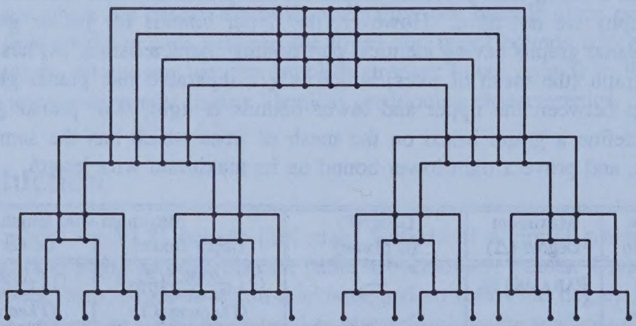


Figure 1: The tree of meshes T_4.

Suppose we wish to limit the number of meshes spanned by an edge to k. Then the vertices of the graph must also be embedded in the internal meshes. This is done by applying the following operation every k levels of the embedding procedure to all the subtrees of meshes at that level. Let G' be the subgraph which is to be embedded in the subtree of meshes T'. Take all the vertices in G' which are incident with previously deleted edges, and 'pull' them up so as to embed them in the root mesh of T'. As a result, all remaining edges incident with these vertices must also be cut, and routed through this mesh and its descendants. For graphs of maximum degree Δ, this may increase the number of edges routed through a mesh by a factor of Δ. However, this does not create many problems since the graph is partitioned using EEBC separators/bifurcators. EEBC dissections create partitions on which previously deleted edges are evenly distributed. Thus the number of additional edges in the meshes at subsequent levels goes down exponentially. If one chooses

$k \geq c \log \Delta$ for some suitably large constant c, the size of the meshes at the time of a pull-up is within a constant factor of their original size.

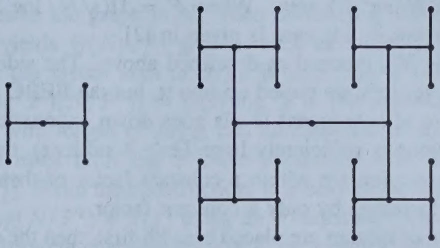


Figure 2: H-Tree layouts of complete binary trees of height 2 and 4.

The inter-mesh wire length is minimized by combining two different layout schemes. The first places meshes in breadth first order as in Figure 1, while the second, called the H-tree scheme, is illustrated for complete binary trees in Figure 2. Take the H-tree layout of a complete binary tree having l^2 leaves, and expand its dimensions by a factor β. The resulting H-channel tree has square vertices of side β, connected by channels of width β. Now mould the breadth-first layout of the given tree of meshes along the channels of H-channel tree until the sizes of the subtrees of meshes are small enough to fit as an H-tree layout of side β. In this way, the long inter-mesh wires closer to the root mesh are shortened at the expense of elongating the shorter ones at subsequent levels. A layout of $T_{cB,2\log n/B}$ using this scheme has $O(B^2 \log^2 n/B)$ area, such that all mesh edges are of unit length, and the longest edge connecting any two meshes has length at most $O(B \log(n/B)/\log\log(n/B))$.

Note that the two stages do not interfere with each other. The first leaves the inter-mesh connections unchanged, and the second leaves the layouts of individual meshes unchanged. Neither stage is effective by itself, but their combination yields layouts of lower maximum wire length. Thus if the maximum edge length between adjacent meshes is not minimized, then the distance between the root mesh and its children, and hence the maximum wire length, is proportional to the side of the layout. If the vertices of the graph are embedded only in the leaf meshes, then adjacent vertices of the graph may end up in leaf meshes lying on opposite ends of the layout.

3 Planar Graphs of Bounded Degree

In light of our layouts of area $O(N \log^2 F)$ for bounded degree planar graphs of gauge F in [8], we establish corresponding gauge-sensitive bounds for the maximum wire length in the theorem below.

Theorem 3.1 *Let G be a planar graph of bounded degree and gauge F. Then G has a layout of area $O(N \log^2 F)$, such that the length of the longest edge is at most*

$$\begin{array}{ll} O(\sqrt{N} \log N/\log\log N) & \text{if} \quad F = \Omega(\sqrt{N}/\log N) \\ O(\sqrt{N} \log F/\log(N/F^2)) & \text{otherwise} \end{array}$$

Proof: For the upper bounds on layout area, see [8]. These are obtained by embedding G in a truncated tree of meshes of depth $\log N$, using a strong $F \log x$ EEBC separator, followed by the $\sqrt{x}$ EEBC separator when it is smaller. At any level i, the resulting tree of meshes has meshes of size $d_i \times d_i$, where $d_i = O(\min(F\log(N/2^i), \sqrt{N/2^i}))$. The resulting layout has $O(N\log^2 F)$ area. When $F = \Omega(\sqrt{N}/\log N)$, the $\sqrt{x}$ separator is smaller. The desired layout in this case is given in [2].

For $F = o(\sqrt{N}/\log N)$, proceed as described above. The side of a mesh increases by a constant factor when vertices are pulled up into it, but the EEBC property of the separator ensures that the increase at subsequent levels goes down exponentially. Thus if the period k of the pull-up operations is sufficiently large ($k \geq 3$ suffices), the sides of the meshes at the time of a pull-up operation are within a constant factor of their original size. The area of the resulting layout increases by only a constant factor.

If L levels of the tree of meshes are placed breadth-first, then the channels of the expanded H-tree must be wide enough to accommodate them. Hence

$$2^L(\log N - L)F = \Theta(\beta)$$

To ensure that the remaining levels of the truncated tree of meshes of G are small enough to fit in the l^2 leaves of the expanded H-tree, we must have $N\log^2 F = \Theta(l^2\beta^2)$, i.e.,

$$\sqrt{N}\log F = \Theta(l\beta)$$

Finally, note that the maximum wire length in the portion embedded in the leaves of the expanded H-tree is $\Theta(\beta)$. The total length from the root of the expanded H-tree to its leaves is $\Theta(l\beta)$. Thus the inter-mesh length, and hence the maximum edge length in the levels embedded breadth-first in the internal channels of the expanded H-tree is $\Theta(l\beta/L)$. Hence a maximum wire length of $\Theta(\beta)$ can be attained if

$$l = \Theta(L)$$

Solving for β from these three relations gives $\beta = O(\sqrt{N}\log F \,/\, \log(N/F^2))$. Hence the maximum wire length is $O(\sqrt{N}\log F \,/\, \log(N/F^2))$. □

From [6], we know that the maximum wire length of complete binary trees is at least $\Omega(\sqrt{N}/\log N)$. Hence the above theorem gives layouts of optimal area and maximum wire length for planar graphs of constant gauge. The following theorem addresses the question for arbitrary F.

Theorem 3.2 *There exist planar graphs of bounded degree and gauge F such that their layout area is $\Omega(N\log F)$, and the length of their longest edge is at least*

$$\begin{array}{ll} \Omega(\sqrt{N\log N}/\log N) & \text{if} \quad F = \Omega(\sqrt{N/\log N}) \\ \Omega(\sqrt{N\log F}/\log N) & \text{otherwise} \end{array}$$

Proof: For graphs of gauge $\Omega(\sqrt{N/\log N})$, the desired lower bound is given in [4]. For $F = o(\sqrt{N/\log N})$, take the graph G consisting of $\Theta(N/F^2\log F)$ disjoint graphs, each of which is a tree of meshes T_F. As shown in [7], area of G is $\Theta(N\log F)$. A counting argument similar to that in [4] shows that there exist two leaves of the T_Fs in G which are a distance $\Omega(\sqrt{N\log F})$ apart in any layout of G. Augment G by constructing a complete binary tree on all its leaves. Then the shortest path between any two leaves has at most $O(\log N)$ edges. Hence the maximum wire length in any layout of this graph is at least $\Omega(\sqrt{N\log F}/\log N)$. □

To exhibit a lower bound for nonplanar graphs having the same partition tree as planar graphs of gauge $F = O(\sqrt{N}/\log N)$, construct a graph G as follows. For some positive constant $c < 1$, arrange cN vertices of the graph in an $F \times (cN/F)$ matrix. Build a complete binary tree over each row and each column. Thus there are F row trees, each having cN/F leaves, and cN/F column trees, each having F leaves. Choose c such that the total number of nodes in the graph is N. Then eliminating the top $\log(N/F^2) + O(1)$ levels of the row trees yields $\Theta(N/F^2)$ graphs, each of which is a mesh of tree $M_{2,F}$. From [4], we know that the layout area of $M_{2,F}$ is $\Omega(F^2 \log^2 F)$. Hence the layout area of G is at least $\Omega((F^2 \log^2 F)(N/F^2)) = \Omega(N \log^2 F)$. The argument in [4] for the lower bound on the maximum wire length of $M_{2,n}$, can be extended to show that for any layout of G which has area at most $O(N \log^2 F)$, the maximum wire length of G is at least $\Omega(\sqrt{N} \log F / \log(N/F^2))$. From the construction, it is easy to see that G can be recursively partitioned up to level $\log(N/F^2) + V(1)$ using an F-separator, and thereafter using a $\sqrt{x}$ separator. Hence the partitioning characteristics of G are identical to those of bounded degree planar graphs of gauge F. Hence for all $F = O(\sqrt{N}/\log N)$, there exist nonplanar graphs having partitioning characteristics identical to planar graphs of gauge F, and for which the area found in [8] and the maximum wire length achieved in Theorem 3.1 are optimal, up to constant factors.

4 Graphs of Arbitrary Maximum Degree

We now consider graphs of arbitrary degree. Again, our strategy will be to minimize the wire length by modifying the our area efficient layouts in [9, 8]. We first consider arbitrary graphs which possess good dissections, and then planar graphs of arbitrary degree and gauge.

4.1 Graphs Partitioned with Separators and Bifurcators

The following theorem considers graphs having strong $(B, \sqrt{2})$ EEBC bifurcators. The result is directly applicable to $\sqrt{x}$ separable graphs. We also consider x^α-separable graphs.

Theorem 4.1 *Let G be an N-vertex graph having a strong $(B, \sqrt{2})$ EEBC bifurcator and maximum degree Δ. Then G has a layout of area $O(B^2 \log^2(N/B))$ and maximum wire length $O\left(B \log(N/B)/ \log_\Delta \log(N/B)\right)$.*

Proof: Consider the above wire-length minimizing strategy. In order to achieve the best possible trade-off between the two stages, the number of meshes k spanned by an edge of the graph must be determined carefully. Recall that choosing $k = c \log \Delta$ for some suitably large constant c ensures that the size of the meshes at the time of a pull-up is within a constant factor of the original size. If the pull-up operations of the first stage are applied every $\Theta(\log \Delta)$ levels over the entire tree of meshes of G, the width/height of the resulting layout is larger by a factor of Δ. However, one can stop earlier. To see this, note that in the second stage, the sub-trees of meshes of G which are embedded in the l^2 leaves of the expanded H-tree are laid out like an H-tree. For a $\beta \times \beta$ size H-tree layout of a tree of meshes of any graph, the length of the longest edge in the graph is only $O(\beta)$. Hence the pull-up operations of the first stage need not be applied to the sub-trees of meshes embedded in the l^2 leaves of the expanded H-tree. As a result, the area of the layout increases only by a constant factor.

In order that the remaining levels of the truncated tree of meshes of G fit in the l^2 leaves of the expanded H-tree, we must have $l^2\beta^2 = \Theta(B^2 \log^2(N/B))$. Hence

$$l\beta = \Theta(B \log(N/B))$$

Let L be the number of levels of the truncated tree of meshes of G which are embedded in breadth-first fashion along the internal channels of the expanded H-tree. The total length from the root of the expanded H-tree to its leaves is $\Theta(l\beta)$. Thus the inter-mesh length in the internal channels of the expanded H-tree is at most $l\beta/L$. Hence the maximum edge length in the internal channels is $\Theta(l\beta \log \Delta / L)$. Thus to achieve a maximum edge length of $\Theta(\beta)$, we must have $\beta = \Theta(l\beta \log \Delta / L)$, i.e.,

$$L = \Theta(l \log \Delta)$$

Finally the channels of the expanded H-tree must be wide enough to accommodate the breadth-first layout of the initial L levels of the truncated tree of meshes of G. To ensure this, it suffices to require β to be large enough to accommodate all the L levels, i.e.,

$$\beta = \Theta(2^{L/2}F)$$

Solving for L, l, and β from the three relations above gives

$$\begin{aligned} l &= \Theta(\log_\Delta \log(N/B)), \qquad L = \Theta(\log\log(N/B)), \quad \text{and} \\ \beta &= \Theta(B \log(N/B) / \log_\Delta \log(N/B)) \end{aligned}$$

Hence the area of the layout is $O(B^2 \log^2(N/B))$, and the length of the longest wire in the layout is at most $O(B \log(N/B) / \log_\Delta \log(N/B))$. □

The lower bound of $\Omega(B \log(N/B) / \log\log(N/B))$ in [1] for bounded degree graphs clearly applies to graphs of arbitrary degree as well. However it is short of the upper bound by a factor of $\log \Delta$. Unfortunately the proof in [1] cannot be readily extended to graphs of arbitrary degree. Hence the above result can be shown to be optimal only when $\Delta = O(1)$.

The above result is directly applicable to $\sqrt{x}$ vertex separable graphs, since they possess a strong $(B, \sqrt{2})$ EEBC bifurcator for $B = O(\Delta\sqrt{N})$. For x^α separable graphs, $\alpha > 1/2$, the maximum wire length cannot be improved. For $\alpha < 1/2$, we have the following result.

Theorem 4.2 *Let G be an N-node graph having a x^α EEBC separator, $0 < \alpha < 1/2$, and maximum degree Δ. Then G has a layout of area $O(\Delta^{1/(1-\alpha)}N)$ and maximum wire length at most $O\left(\sqrt{\Delta^{1/(1-\alpha)}N} / \log_\Delta N\right)$. Moreover, there exist graphs having Δx^α edge separators for which these bounds are optimal.*

Proof: The desired layouts are found by transforming the layouts of these graphs in [9] along the lines of Theorem 4.1. Their optimality is demonstrated by a Δx^α edge separable graph called the *augmented tree of superconcentrators*. We confine our discussion here only to its description. The graph is based on the *tree of superconcentrators* T^E_{Δ,x^α}. T^E_{Δ,x^α} is a graph resembling a complete binary tree of depth $\log(N/\Delta^{1/(1-\alpha)}) + O(1)$, whose nodes have been replaced by superconcentrators, and edges by sets of edges between the superconcentrators which replaced their end points. The nodes at level $i \geq 0$ are replaced by linear size superconcentrators having $\Delta(N/2^i)^\alpha$ inputs and $\Delta(N/2^{i+1})^\alpha$ outputs. The outputs of a superconcentrator at level i are connected to the corresponding inputs of its two neighboring superconcentrators at level $i+1$. The *augmented tree of superconcentrators* is the graph formed by building a complete Δ-ary tree on the leaf nodes of T^E_{Δ,x^α}. □

4.2 Planar Graphs of Arbitrary Degree and Gauge

We have described gauge and degree sensitive layouts of planar graphs in [8]. The following theorem applies the above strategy to minimize the maximum wire length in these layouts as well.

Theorem 4.3 *Let G be an N-vertex planar graph of maximum degree Δ and gauge F. Then G has a layout having the following area and maximum wire length respectively:*

$$O(\Delta N \log^2(N/\Delta)) \quad \text{and} \quad O\left(\sqrt{\Delta N}\log(N/\Delta)/\log_\Delta \log N\right) \quad \text{if } \Delta F = \Omega(\sqrt{\Delta N}/\log N)$$
$$O(\Delta N \log^2 F) \quad \text{and} \quad O\left(\sqrt{\Delta N}\log F/\log_\Delta (N/F^2)\right) \quad \text{otherwise}$$

Proof: The upper bounds on layout area follow from [8]. These are obtained by embedding the graph in a truncated tree of meshes of depth $\log(N/\Delta)$ by using a strong $(\Delta F)\log x$ EEBC separator for the initial partitioning, and switching to a strong $\sqrt{\Delta x}$ EEBC separator when it yields smaller cuts. Thus the meshes at any level i are of size $d_i \times d_i$, where $d_i = O(\min(\Delta F)\log(N/2^i),\ \sqrt{\Delta N/2^i}))$.

The maximum wire length for the first case follows from Theorem 4.1, since the strong $\sqrt{\Delta N}$ EEBC separator for planar graphs is the same as a strong $(c\sqrt{\Delta N}, \sqrt{2})$ EEBC bifurcator for some constant c. For the other case, i.e., $\Delta F \log N = O(\sqrt{\Delta N})$, proceed along the same lines as Theorem 4.1. Then the relationships between β, l, L, F, Δ, and N are:

$$\begin{aligned} 2^L(\log N - L)\Delta F &= \Theta(\beta) \\ \sqrt{\Delta N}\log F &= \Theta(l\beta) \\ l &= \Theta(L/\log\Delta) \end{aligned}$$

Solving for β from these three relations gives

$$\beta = O\left(\sqrt{\Delta N}\log F \log\Delta \,/\, \log(\sqrt{\Delta N}/(\Delta F))\right) = O(\sqrt{\Delta N}\log F \,/\, \log_\Delta(N/F^2))$$

Since the maximum wire length in the layout is $O(\beta)$, the result follows. □

The above results are optimal for planar graphs of constant gauge.

Theorem 4.4 *For every value of Δ and N, there exists an N-node planar graph of constant gauge and maximum degree Δ for which the layouts produced by Theorem 4.3 have optimal area and maximum wire length.*

Proof: In [6], it is shown that any layout of an N-node complete binary tree has a wire of length $\Omega(\sqrt{N}/\log N)$. It is easy to extend the argument to show that any layout of an N-node complete Δ-ary tree has $\Omega(\Delta N)$ area and maximum wire length $\Omega(\sqrt{\Delta N}/\log_\Delta N)$. Thus for graphs of constant gauge, the layouts produced by the above theorem are optimal in area and maximum wire length. □

For the remaining cases, the following theorem establishes lower bounds which fall short of the upper bounds, but demonstrate a gap which is a function of F and Δ. This gap corresponds to the one in Section 3 for planar graphs of bounded degree.

Theorem 4.5 *For every value of Δ and F, there exists an N-node planar graph of gauge F and maximum degree Δ for which any layout requires at least the following area and maximum wire length respectively:*

$$\Omega(\Delta N \log(N/\Delta)) \quad \text{and} \quad \Omega\left(\sqrt{\Delta N \log(N/\Delta)}/\log_\Delta N\right) \quad \text{if } \Delta F = \Omega(\sqrt{\Delta N}/\log N)$$
$$\Omega(\Delta N \log F) \quad \text{and} \quad \Omega\left(\sqrt{\Delta N \log F}/\log_\Delta N\right) \quad \text{otherwise}$$

Proof: The desired lower bounds can be derived by extending the area lower bounds in [7], which were exhibited by the Δ-modified tree of meshes T_n^Δ. T_n^Δ is essentially a truncated tree of meshes $T_{n,2\log(n/\Delta)+4}$ in which sub-meshes of size $\Delta/4 \times \Delta/4$ in every mesh are replaced by vertices of degree Δ, in such a way that every vertex of degree Δ is surrounded by a Δ vertices of degree 4. $T_{n,\Delta}$ has $\Theta(n^2/\Delta)$ vertices of degree 4 and $\Theta(n^2/\Delta^2)$ leaves, each of which is a vertex of degree Δ. The total number of nodes in T_n^Δ is $N = \Theta(n^2 \log(n/\Delta)/\Delta)$ and its area is $\Theta(n^2 \log^2(n/\Delta)) = \Theta(\Delta N \log(N/\Delta))$. Construct a complete Δ-ary tree with the leaves of T_n^Δ as its leaves. The number of new nodes added to T_n^Δ is only $O(n^2/\Delta^2)$. The resulting graph is called the *augmented* Δ-modified tree of meshes, or *augmented* T_n^Δ. Using a counting argument similar to the one in [4], it can be shown that any layout of the augmented T_n^Δ contains two leaves which are at least $\Omega(n \log(n/\Delta))$ apart. Since the shortest path between any two leaves has at most $4\log_\Delta(n/\Delta)$ edges, it follows that there is an edge of length at least $\Omega(n\log(n/\Delta)/\log_\Delta(n/\Delta))$. Expressed in terms of the number of nodes N, there is an edge of length $\Omega(\sqrt{\Delta N \log(N/\Delta)} / \log_\Delta N)$.

The gauge of T_n^Δ is $O(n/\Delta) = O(\sqrt{N/(\Delta\log(N/\Delta))})$. For smaller values of the gauge F, take $\Theta(N/(\Delta F^2 \log F))$ graphs, each of which is a $T_{\Delta F}^\Delta$. Construct a complete binary tree with the $\Theta(N/(\Delta \log F))$ leaves of the $T_{\Delta F}^\Delta$s as its leaves. Again, it can be shown that there are two leaves of this graph which are $\Omega(\sqrt{\Delta N \log F})$ apart. Since there is a path of length at most $O(\log_\Delta(N/(\Delta\log F))) = O(\log_\Delta N)$ between any two leaves, it follows that one of these edges has length at least $\Omega(\sqrt{\Delta N \log F}/\log_\Delta N)$. □

As with Section 3, it is possible to demonstrate tighter lower bounds on the maximum wire length for nonplanar graphs which have the same partitioning characteristics as the planar graphs above. However, as in Section 4, these lower bounds fall short of the above upper bounds by a factor of $\log \Delta$.

5 Concluding Remarks

We presented general techniques to minimize the maximum wire length in layouts of graphs of arbitrary degree, and measured their optimality by establishing corresponding lower bounds. For graphs partitioned using bifurcators, our layouts have optimal area, and the difference between our upper and lower bounds on the maximum wire length is logarithmic in Δ. For planar graphs our bounds on the maximum wire length are sensitive to the maximum degree, and span the spectrum between outerplanar graphs and arbitrary planar graphs. Our techniques for planar graphs are optimal for graphs of constant gauge. In other cases, the gaps between our upper and lower bounds are a function of the gauge F and Δ, and correspond to the gap in [4] for bounded degree planar graphs.

Our methods rely only on the recursive partitioning characteristics of the underlying graph. Thus they also work for nonplanar graphs having identical recursive partitioning characteristics. As in [4], it is possible to show that our methods are more effective for corresponding nonplanar graphs which have the same partitioning characteristics. The lower bounds we established to this end demonstrate the optimality of our techniques for bounded degree graphs over almost the entire range of values of the gauge. For values of the gauge between $\sqrt{N}/\log N$ and $\sqrt{N/\log N}$, our method is suboptimal. We have recently discovered a way to remedy this anomaly. For arbitrary degree graphs having $\sqrt{x}$ separators or bifurcators, our methods yield optimal area, but our lower bounds on the maximum wire length are still short by a factor of $\log \Delta$. However, except for $\alpha = 1/2$, we achieve optimal area and wire length for x^α separable graphs, $0 \leq \alpha \leq 1$. Thus our results show that graph dissection based layout schemes work well over a wide range of graphs, and even when the usual restrictions on vertices of bounded degree are removed.

Finally, note that our results express bounds for the worst case. Thus a planar graph of maximum degree Δ is assumed to have $\Theta(N/\Delta)$ vertices of degree Δ. However, as we did in [8], similar upper and lower bounds can also be derived for graphs which have fewer vertices of high degree. We have also not addressed the issue vertices of arbitrary area. However, our representation of vertices of arbitrary degree by appropriate regions makes such an extension straightforward.

References

[1] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28:300–343, 1984.

[2] S. N. Bhatt and C. E. Leiserson. Minimizing the longest edge in a VLSI layout. Technical report, MIT Lab. for Computer Science, 1981.

[3] D. Dolev, F. T. Leighton, and H. W. Trickey. Planar embedding of planar graphs. In *Advances in Computing Research, vol. 2*, pages 147–161. JAI Press, 1984.

[4] F. T. Leighton. New lower bound techniques for VLSI. *Proc. 22nd Annual Symposium on Foundations of Computer Science*, pages 1–12, 1981.

[5] C. E. Leiserson. *Area Efficient VLSI Computation.* PhD thesis, Carnegie Mellon University, 1981.

[6] M. S. Paterson, W. L. Ruzzo, and L. Snyder. Bounds on minimax edge length for complete binary trees. *Proc. 13th Annual ACM Symposium on Theory of Computing*, pages 293–299, 1981.

[7] D. D. Sherlekar. Optimality of gauge and degree-sensitive VLSI layouts of planar graphs. In *Selected Papers from the Int'l Conference on Computing and Informatics.* Lecture Notes in Computer Science, Springer Verlag, June 1990.

[8] D. D. Sherlekar and J. JáJá. Input sensitive VLSI layouts of graphs of arbitrary degree. *Proceedings of the Third International Agean Worshop on Computing*, pages 268–277, June 1988.

[9] D. D. Sherlekar and J. JáJá. Layouts of graphs of arbitrary degree. *Proceedings of the 25th Annual Allerton Conference*, pages 355–364, September 1987.

[10] C. D. Thompson. *A Complexity Theory for VLSI.* PhD thesis, Carnegie Mellon University, 1980.

[11] L. G. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, C-30(2):135–140, Feb 1981.

Routing with Short Wires
and
Small Channel Width

Dinesh Bhatia*
Department of Computer Science and Engineering
School of Engineering and Applied Science
Southern Methodist University
Dallas, Texas 75275-0122

Abstract

In this paper we address the problem of routing two terminal nets on a two dimensional $N \times N$ grid. We assume that each net length is bounded by k, where k is a small constant or a small function of N $(k \ll N)$. The problem arises during the wiring of reconfigured wafer scale arrays. We establish bounds for the worst case channel width needed to route an $N \times N$ two dimensional grid. We develop an $O(N^2 k \ \log \ k)$ sequential algorithm to route wires in a one-bend fashion with channel width at most $\frac{3}{2}k + 4$. We also give a parallel realization of the algorithm stated above that runs in $O(k^3 \cdot \log k)$ parallel time. Our results perform better than all the existing work on the similar problem.

1. Introduction

The problem of routing two dimensional grids using short wires arises during the wiring of the reconfigured wafer scale arrays and some restricted cases of global routing of gate arrays. In the *reconfiguration* problem, good workable systems are derived on a single faulty wafer by producing an interconnection topology for good elements. The reconfiguration problem for two-dimensional arrays has been studied quite extensively by [1], [4] and others. Following the reconfiguration, the good elements (chips) have to be connected during a phase called *restructuring* . In the restructuring of wafers, wires are drawn in rectilinear channels to connect the good elements.

The usual parameters for efficient routing are the cost of restructuring and the resulting width of the wiring channels. Therefore, before any restructuring is done, the fundamental issue of efficient routing must be considered (i.e., wiring with small channel widths). Also, for the reliability of restructuring, it is important to have a minimum number of bends per wire.

2. Model and Nomenclature

Routing is performed on a regular two-dimensional $N \times N$ array $G(V, E)$, where nodes represent elements of the wafer and the edges represent the channels through which wires can be routed. A *net* is a set of nodes to be connected. In this paper, we consider only two-terminal nets. Each node is mapped by at most one terminal. A net consisting of nodes u and v will be represented by $[u, v]$. A *wire* is a set of edges that connect the nodes of a net. Each wire is of length at most k, where k is a small constant or a small function of N. The *width* of an edge is the number of wires that contain that edge. The *channel width* is the maximum width taken over all the edges of the $N \times N$ array. In addition, we define $C_t^k(N)$ as the channel width for routing an $N \times N$ array when the length of each wire is at most k and each wire has at most t bends. Consider a set of

---

* Dinesh Bhatia was supported by ACM-SIGDA Design Automation Scholarship

m consecutive nodes $v_1, v_2, \cdots, v_m$ in a row (column) of the $N \times N$ array. $W_t^k(m)$ is the number of horizontal (vertical) wires that pass over the edges between v_1 and v_m when the length of each wire is at most k and each wire has at most t bends.

We call the routing problem as *restricted* if each net length is bounded by k. We call the routing problem as *generalized* if there exists no restriction on the length of the nets.

3. Past Work

Karp et.al. studied the *generalized* problem in connection with the global routing for gate arrays on a regular $N \times N$ grid and established tight bounds on channel width. They also developed provably good heuristics and approximation techniques to carry out routing. Karp et. al. showed how to find one-bend routing on an $N \times N$ array with channel width at most $\lceil \frac{N}{2} \rceil + 1$. They also showed how to find a solution of width $C\ O(log\,\frac{N}{C})$, where C is the optimal channel width for an instance. This approximation result has previously been obtained by Burstein and Pelavin [2]. Raghavan and Thompson have described provably good approximation algorithm that guarantees a channel width of $C + \sqrt{C\,log\,N}$ with high probability [5]. Saito et.al. [Sa] studied problem of routing on grid in context with routing with in channels. General routing problem has been studied extensively by many researchers like [6],[2].

In the presence of all the tight results, the routing problem for wafers is not addressed properly as most wires are very short, typically of the $O(log\,N)$. In fact, it has been shown in [1] that with high probability, the wires are of length at most $O(log^{\frac{3}{4}} N)$ and for 50 % faults distributed uniformly over the wafer, the constant before the logarithmic term is 1.0. Thus in the presence of such short wires, all of the previous work would yield channel width that is also the worst case upper bound, i.e., the maximum attainable value. We start with a very basic result.

Theorem 1: For an $N \times N$ two dimensional grid, any instance of upto $\lfloor N^2/2 \rfloor$ nets can be routed with channel width at least k and at most $2k - 2$ $(k \leq \lfloor N/2 \rfloor + 2)$, where k is the length of the longest net. In other words,

$$k \ \leq \ C_1^k(N) \ \leq \ 2k - 2, \ \ (k \ \leq \ \lfloor N/2 \rfloor \ + \ 2)$$

Proof:

The worst case lower bound is easy to establish by construction. Consider an $N \times N$ grid. For any row, we connect nodes $\{\ (i,j),\ (i,j+1),\ (i,j+2),\ \ldots\ (i,j+k-1)\ \}$ to nodes $\{\ (i,j+k),\ (i,j+k+1),\ (i,j+k+2),\ \ldots\ (i,j+2k-1)\ \}$ respectively. It is easy to see that the width between $(i,j+k-1)$ and $(i,j+k)$ is k. Figure 1 shows such construction for $k\ =\ 5$.

For upper bound we try to maximize the width of channel in a greedy way. Consider any two successive nodes (i,j) and $(i,j+1)$. Any wire that passes over the edge between nodes (i,j) and $(i,j+1)$ must have one of its two terminals as a node in row i as we are routing in one bend fashion. Note that there exist k nodes to the right of node (i,j) which are at distance at most k from (i,j). Also there exist at most k nodes to the left of $(i,j+1)$ that are at distance at most k from $(i,j+1)$. All the wires starting at these nodes can pass through the section between (i,j) and $(i,j+1)$ to make total width of $2k$. We note that the extreme right and left nodes can be connected to two nodes in the same row i as the length is at most k. Thus the width can at most be $2k-2$. It is easy to see that other combinations also yield width at most $2k-2$. Figure 2 illustrates the upper bound example for the case when $k = 5$.

After proving these bounds, the obvious question that comes is if it is possible to route wires with the channel widths less than $2k-2$. We show that, there exists a way of routing wires with channel width at most $(\frac{3}{2}k+4)$. The bounds are still stronger due to the following statement :

There exists a way of routing wires such that the number of horizontal (vertical) wires over k consecutive nodes in a row (column) is atmost $(\frac{3}{2}k+4)$. The k consecutive nodes are predefined and are not chosen arbitrarily.

Before we proceed any further, we state the following result in the form of a theorem. The proof of theorem is omitted and can be found in [3].

Theorem 2 (Rounding Theorem [3])

Let A be a real-valued $r \times s$ matrix, let x be a real-valued s vector, let b be a real-valued r vector such that $Ax = b$, and let Δ be a positive real number such that in every column of A,

(i) the sum of the positive elements is $\leq \Delta$, and

(ii) the sum of the negative elements is $\geq -\Delta$. Then we can compute an integral s vector $\hat{x}$ such that

(i) for all i, $1 \leq i \leq s$, either $\hat{x}_i = \lfloor x_i \rfloor$ or $\hat{x}_i = \lceil x_i \rceil$, and

(ii) $A\hat{x} = \hat{b}$, where $\hat{b}_i - b_i < \Delta$ for $1 \leq i \leq r$. In the case that all entries in **A** are integers, then a stronger bound applies : $\hat{b}_i - \lceil b_i \rceil \leq \Delta - 1$. If **x** contains d distinct components, we obtain the integral approximation $\hat{x}$ in time $O(r^3 log(1 + s/r) + r^3 + d^2r + sr)$.

We use rounding theorem by formulating the one-bend routing problem as an integer program and a $0-1$ valued variable x_i is associated with each net n_i. The values 0 and 1 will represent one of two possible **L** shaped routes taken by the routing. The convention for $0-1$ assignment is shown in Figure 3. The $0-1$ valued variables associated with each of $\lfloor N^2/2 \rfloor$ nets could be used to find the total number of wires in each row and column. An example for routing using the $0-1$ assignment to the variable x_i is shown in Figure 4. The number of wires, for example, in row 3 is given by $(1-x_4)+x_3+(1-x_1)+x_6$ which is also equal to $(x_3+x_6)-(x_1+x_4)+2$. In general, the number of wires in a row (column) can be written as $\sum x_i - \sum x_j + c_1, 1 \leq i,j \leq \lfloor \frac{N^2}{2} \rfloor$.

4. Routing

We perform routing in four phases. In the first phase, we divide the $N \times N$ grid into subgrids of size $2k \times 2k$ each. This is illustrated in Figure 5. For convenience, we assume that $N \bmod 2k = 0$. However, the general case of $N \bmod 2k \neq 0$ is not difficult to route using same scheme described below. Also, for the sake of succintness, we give analysis for k consecutive nodes in the row (column) of the $k \times k$ subgrid A_1 in Figure 6. Similar analysis holds for all the $k \times k$ subgrids shown in Figure 6.

4.1 Phase 1 of Routing

Each subgrid of size $2k \times 2k$ is routed using the rounding result due to Karp et.al. We associate a $0-1$ valued variable x_i with each net n_i where $1 \leq i \leq (2k^2 - \frac{y}{2})$. y is the number of nets that have one terminal in the $2k \times 2k$ subgrid under consideration and other in one of its adjacent subgrids. Terminals of the net that run between two adjacent $2k \times 2k$ subgrids are called *forbidden* terminals. A terminal is called *routed* if the net to which it belongs has been routed previously. Clearly, from our definition, a $2k \times 2k$ subgrid under consideration will have y *forbidden* terminals in Phase 1. Nets comprising of *forbidden* and *routed* terminals are not routed in the current phase.

To route the $2k \times 2k$ subgrid using rounding theorem, we use the following setup of equations:

$$Ax = b$$

where A is a $4k \times (2k^2 - \frac{y}{2})$ integer-valued matrix, x is a $(2k^2 - \frac{y}{2})$ vector which is set to $(\frac{1}{2}, \frac{1}{2}, \cdots, \frac{1}{2})^T$ and whose rounding will give a $0-1$ assignment to the variables associated with the nets. b is a $4k$ vector which is set to $(k-\delta_1(\frac{1}{2}), k-\delta_2(\frac{1}{2}), ..., k-\delta_{2k}(\frac{1}{2}), \cdots, k-\delta_{4k}(\frac{1}{2}))^T$ where δ_i $(1 \leq i \leq 2k)$ is the number of *forbidden* terminals in row i of the $2k \times 2k$ subgrid, and δ_i $(2k < i \leq 4k)$ is the number of *forbidden* terminals in the column i of the $2k \times 2k$ subgrid. This is illustrated in Figure 5. The entries of A are as follows :
Each net n_i can take one of the two possible **L** shaped routes. This changes wire count in at most two rows and at most two columns. Also, due to our adopted convention, at most two constraints associated with net n_i are entered positively and at most two constraints are entered negatively. Thus each column of A could have at most two positive and two negative entries and each row of A could have at most $2k$ entries (positive or negative). Thus Δ for any column of A is at most 2.

With this setup for rounding and following the rounding theorem, we compute a $0-1$ valued $(2k^2 - \frac{y}{2})$ vector $\hat{x}$ such that :

$$A\hat{x} = \hat{b}, \text{ where } \hat{b}_i - b_i < \Delta,\ 1 \leq i \leq 4k$$
$$\text{or } \hat{b}_i \leq b_i + \Delta - 1$$

In other words, a solution in which each row i of the $2k \times 2k$ subgrid has no more than $\lceil k - \delta_i(\frac{1}{2}) + 1 \rceil$, $(1 \leq i \leq 2k)$ wires and each column i of the $2k \times 2k$ subgrid has no more than $\lceil k - \delta_i(\frac{1}{2}) + 1 \rceil$, $(2k < i \leq 4k)$ wires.

After the termination of the phase 1 of subgrid routing, the nets that have both terminals with in the subgrid are routed successfully. However, the nets consisting of *forbidden* terminals remain unrouted. The distribution of such unrouted nets is shown in Figure 6. In Figure 6, we divide the $N \times N$ grid into small subgrids of size $k \times k$ each.

By a symbolic representation using a line between two adjacent subgrids S_1 and S_2 we show the possible existence of net(s) with one terminal in S_1 and other in S_2. Thus a line between A_1 and A_2 implies a possible net with one terminal in A_1 and another in A_2. Similarly between A_1 and A_3 and so on. Also, we note from Figure 7 that for each $k \times k$ subgrid, there are five types of wires. For example, consider subgrid B_1. Wires from a terminal in B_1 to a terminal in C_2 are of type 1, from a terminal in B_1 to a terminal in B_4 are of type 2, from a terminal in B_1 to a terminal in B_3 are of type 3, from a terminal in B_1 to a terminal B_2 are of type 4 and finally from a terminal in B_1 to a terminal in A_4 are of type 5. The numbering is shown in Figure 7. We donot develop a formal convention for numbering the types and use the one shown in Figure 7. Also, we define $\delta_{i,B_{ab}}$ to the number of wires in row (column) i of the $k \times k$ subgrid B_a, where $1 \leq a \leq 4$, and of type b , where $1 \leq b \leq 5$. Similar definition holds for all possible $k \times k$ subgrids.

In the following phases of routing, the idea is to route the unrouted nets without disturbing the nets which have been routed in the previous phases.

4.2 Phase 2 of Routing

In the second phase of routing, we route the unrouted nets in the grid of size $(N-2k) \times (N-2k)$. The $(N-2k) \times (N-2k)$ grid is constructed from rows $(k+1)$ to $(N-k)$ and from columns $(k+1)$ to $(N-k)$. The shaded region in Figure 6 shows the $(N-2k) \times (N-2k)$ grid. The routing procedure is identical to the one for Phase 1. We use the following setup of equations:

$$Ax \ = \ b$$

where A is a $4k \times (2k^2 - \frac{y}{2})$ integer-valued matrix, x is a $(2k^2 - \frac{y}{2})$ vector which is set to $(\frac{1}{2}, \frac{1}{2}, \cdots, \frac{1}{2})^T$. b is a $4k$ vector which is set to $(k-\mu_1(\frac{1}{2}), k-\mu_2(\frac{1}{2}), \cdots, k-\mu_{2k}(\frac{1}{2}), \cdots, k-\mu_{4k}(\frac{1}{2}))^T$

where μ_i $(1 \le i \le 2k)$ is the number of *forbidden* and *routed* terminals in row i of the $2k \times 2k$ subgrid. To evaluate μ_i, we consider row i of the $k \times k$ subgrids A_1 and A_2.

The number of terminals in row i of A_1 that have been routed (*routed* terminals) during previous phase is given by $(k - \delta_{i,A_{11}} - \delta_{i,A_{12}} - \delta_{i,A_{13}} - \delta_{i,A_{14}} - \delta_{i,A_{15}})$. Also $(\delta_{i,A_{11}} + \delta_{i,A_{15}})$ number of terminals in row i of A_1 are still *forbidden* . Thus total number of *forbidden* and *routed* terminals in row i of A_1 is $(k - \delta_{i,A_{12}} - \delta_{i,A_{13}} - \delta_{i,A_{14}})$. Similarly for row i of A_2 the total number of *forbidden* and *routed* terminals is $(k - \delta_{i,A_{22}} - \delta_{i,A_{23}} - \delta_{i,A_{24}})$. Note that $\delta_{i,A_{12}} \neq \delta_{i,A_{22}}$. Thus μ_i is given by :

$$\mu_i = (k - \delta_{i,A_{12}} - \delta_{i,A_{13}} - \delta_{i,A_{14}}) + (k - \delta_{i,A_{22}} - \delta_{i,A_{23}} - \delta_{i,A_{24}}).$$

$$\text{or } \mu_i = (2k - \delta_{i,A_{12}} - \delta_{i,A_{13}} - \delta_{i,A_{14}} - \delta_{i,A_{22}} - \delta_{i,A_{23}} - \delta_{i,A_{24}}).$$

Similarly μ_i $(2k < i \le 4k)$ is the number of *forbidden* and *routed* terminals in column i of the $2k \times 2k$ subgrid. The A matrix is filled in the same manner as in Phase 1. A satisfies the condition for rounding theorem for $\Delta = 2$.

On applying the rounding theorem on the above setup of equations, we obtain a $0-1$ valued $(2k^2 - \frac{y}{2})$ vector $\hat{x}$ such that,

$$A\hat{x} = \hat{b} \text{ where } \hat{b}_i - b_i < \Delta, \ (1 \le i \le 4k)$$

$$\text{or } \hat{b}_i < b_i + \Delta$$

$$\text{or } \hat{b}_i \le b_i + \Delta - 1$$

In other words, a solution in which each row i of the $2k \times 2k$ subgrid has no more than $\lceil (k - \mu_i(\frac{1}{2}) + 1) \rceil$ $(1 \le i \le 2k)$ wires and each column i of the $2k \times 2k$ subgrid has no more than $\lceil (k - \mu_i(\frac{1}{2}) + 1) \rceil$ $(2k < i \le 4k)$ wires. This is the partial number of wires through row i of the $k \times k$ subgrid A_1. The total number of wires through row i of the $k \times k$ subgrid A_1 is given by the sum of the partial solution obtained in the second phase and the number of wires through row i of the $k \times k$ subgrid due to previous phase. To evaluate the total number of wires consider row i of $k \times k$ subgrid A_1. The the total number of wires in row i is at most

$$\{k + 1 - \delta_i(\tfrac{1}{2})\} + \{k + 1 - \mu_i(\tfrac{1}{2})\}$$

Also,

$$\delta_i = (\delta_{i,A_{11}} + \delta_{i,A_{12}} + \delta_{i,A_{13}} + \delta_{i,A_{14}} + \delta_{i,A_{15}} + \delta_{i,B_{11}} + \delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}} + \delta_{i,B_{15}})$$

and

$$\mu_i = (2k - \delta_{i,A_{12}} - \delta_{i,A_{13}} - \delta_{i,A_{14}} - \delta_{i,A_{22}} - \delta_{i,A_{23}} - \delta_{i,A_{24}})$$

Thus the total number of wires in row i of A_1 at the end of phase two of routing is given by

$$(k+2) - \tfrac{1}{2}(\delta_{i,A_{11}} + \delta_{i,A_{15}} + \delta_{i,B_{11}} + \delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}} + \delta_{i,B_{15}} - \delta_{i,A_{22}} - \delta_{i,A_{23}} - \delta_{i,A_{24}})$$

After termination of the phase 2 for subgrid routing, the distribution of unrouted nets is illustrated in Figure 8.

4.3 Phase 3 of Routing

In the third phase of routing, we route the unrouted nets in the grid of size $(N-2k) \times N$. The $(N-2k) \times N$ grid is constructed from rows $(k+1)$ to $(N-k)$ and from columns 1 to N . Ths shaded region in Figure 8 shows the $(N-2k) \times N$ grid. The routing is performed using the rounding theorem and is quite similar to the one described for phase two of subgrid routing in section 4.2. We use the following setup of equations:

$$Ax = b$$

Where A and x are respectively the $4k \times (2k^2 - \frac{y}{2})$ integer-valued matrix and a $(2k^2 - \frac{y}{2})$ vector set to $(\frac{1}{2}, \frac{1}{2}, \cdots, \frac{1}{2})^T$. b is a $4k$ vector which is set to $(k - \rho_1(\frac{1}{2}), k - \rho_2(\frac{1}{2}), \cdots, k - \rho_{2k}(\frac{1}{2}), \cdots, k - \rho_{4k}(\frac{1}{2}))^T$, where ρ_i $(1 \le i \le 2k)$ is the number of *forbidden* and *routed* terminals in the row i of $(2k \times 2k)$ subgrid. we eliminate the steps for evaluating ρ_i. This can be found using row i of the $k \times k$ subgrids B_1 and A_1 .

$$\rho_i = (k - \delta_{i,A_{15}}) + (k - \delta_{i,B_{15}})$$
$$\text{or } \rho_i = (2k - \delta_{i,A_{15}} - \delta_{i,B_{15}})$$

Similarly ρ_i $(2k < i \leq 4k)$ is the number of *forbidden* and *routed* terminals in column i of the $2k \times 2k$ subgrid. The entries of A are similar to the ones shown in section 4.2 during phase two of subgrid routing. On applying rounding theorem on the above setup of equations, we obtain a $0-1$ valued $(2k^2 - \frac{y}{2})$ vector $\hat{x}$ such that:

$$A\hat{x} = \hat{b}$$
$$\text{where } \hat{b}_i - b_i \leq \Delta - 1$$

In other words, a solution to subgrid routing in which each row i of the $2k \times 2k$ subgrid has no more than $\lceil (k - \rho_i(\frac{1}{2}) + 1) \rceil$ $(1 \leq i \leq 2k)$ wires and each column i of the $2k \times 2k$ subgrid has no more than $\lceil (k - \rho_i(\frac{1}{2}) + 1) \rceil$ $(2k < i \leq 4k)$ wires . The sum of this partial solution obtained in the third phase of routing and the number of wires passing through row i of A_1 due to previous phases would yield total number of wires through any row i of A_1. This sum is given by

$$(k+3) - \tfrac{1}{2}(\delta_{i,A_{11}} + \delta_{i,B_{11}} + \delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}} - \delta_{i,A_{22}} - \delta_{i,A_{23}} - \delta_{i,A_{24}})$$

The distribution of the unrouted nets at the end of the third phase of routing is illustrated in Figure 9.

4.4 Phase 4 of Routing

In the fourth and the last phase of subgrid routing, we successfully route the unrouted nets and complete the routing over the $N \times N$ grid. In the fourth phase of routing, the unrouted nets in the grid of size $N \times (N - 2k)$ are routed. The $N \times (N - 2k)$ grid is constructed from rows 1 to N and from columns $k+1$ to $N - 2k$. Ths shaded region in Figure 9 shows the $N \times (N - 2k)$ grid. The routing is performed using the rounding theorem and is similar to the ones described for previous phases. The setup of equations is as follows :

$$Ax = b$$

Where A and x are respectively the $4k \times (2k^2 - \frac{y}{2})$ integer-valued matrix and a $(2k^2 - \frac{y}{2})$ vector set to $(\frac{1}{2}, \frac{1}{2}, \cdots, \frac{1}{2})^T$. b is a $4k$ vector which is set to $(k - \gamma_1(\frac{1}{2}), k - \gamma_2(\frac{1}{2}), \cdots, k - \gamma_{2k}(\frac{1}{2}), \cdots, k - \gamma_{4k}(\frac{1}{2}))^T$, where γ_i $(1 \leq i \leq 2k)$ is the number of *forbidden* and *routed* terminals in the row i of $(2k \times 2k)$ subgrid. γ_i can be evaluated using row i of the $k \times k$ subgrids A_1 and A_2 .

$$\gamma_i = (k - \delta_{i,A_{11}}) + (k - \delta_{i,A_{21}})$$
$$\text{or } \gamma_i = (2k - \delta_{i,A_{11}} - \delta_{i,A_{21}})$$

Similarly γ_i $(2k < i \leq 4k)$ is the number of *forbidden* and *routed* terminals in column i of the $2k \times 2k$ subgrid. The entries of A are similar to the ones shown in section ** during phase two of subgrid routing. On applying rounding theorem on the above setup of equations, a $0-1$ valued $(2k^2 - \frac{y}{2})$ vector $\hat{x}$ is obtained such that

$$A\hat{x} = \hat{b}$$
$$\text{where } \hat{b}_i - b_i \leq \Delta - 1$$

In other words, a solution to subgrid routing in which each row i of the $2k \times 2k$ subgrid has no more than $\lceil (k - \gamma_i(\frac{1}{2}) + 1) \rceil$ $(1 \leq i \leq 2k)$ wires and each column i of the $2k \times 2k$ subgrid has no more than $\lceil (k - \gamma_i(\frac{1}{2}) + 1) \rceil$ $(2k < i \leq 4k)$ wires . The sum of this partial solution obtained in the third phase of routing and the number of wires passing through row i of A_1 due to previous phases would yield total number of wires through any row i of A_1. This sum is given by

$$(k+4) - \tfrac{1}{2}(\delta_{i,B_{11}} + \delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}} - \delta_{i,A_{21}} - \delta_{i,A_{22}} - \delta_{i,A_{23}} - \delta_{i,A_{24}})$$

Note that, in one-bend routing, the nets starting from the terminals $(\delta_{i,B_{11}} + \delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}})$ do not pass through the rows of A_1 . Thus there is no effect on the channel width in A_1 due to the terminals $(\delta_{i,B_{11}} + \delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}})$. Also, $(\delta_{i,A_{21}} + \delta_{i,A_{22}} + \delta_{i,A_{23}} + \delta_{i,A_{24}} + \delta_{i,A_{25}}) \leq k$. Thus the total number of horizontal (vertical) wires in the row (column) of A_1 is at most $(\frac{3}{2}k+4)$.

From the above analysis, we show that the channel width is at most $(\frac{3}{2}k+4)$. In the analysis, all the edges within the $k \times k$ subgrids were taken into account. However, it is important to note that there are edges that join a $k \times k$ subgrid to its four adjacent $k \times k$ subgrids. In all four phases, we never checked the width of these edges. In the following discussion, the width (number of wires over the edge) of such edges is analysed. The width of the channel is analysed for the edges that run between the subgrids A_1 and A_2 , A_1 and A_4 , A_1 and B_1 , and B_1 and C_1 . The analysis between these boundaries is necessary and sufficient as each boundary is one of the 4 types.

The number of wires that pass over any edge is the sum total of the number of wires that pass over the same edge in each of the four phases. In the table below, we summarise the number of wires that pass over an edge in each phase and the total number of wires that would pass finally. A wire is called *redundant* if it never passes over the edge under consideration.

	Edge on Boundary Between A_1 and A_2
Width after Phase 1	0
Width after Phase 2	$1 + \frac{1}{2}(\delta_{i,A_{12}} + \delta_{i,A_{13}} + \delta_{i,A_{14}} + \delta_{i,A_{22}} + \delta_{i,A_{23}} + \delta_{i,A_{24}})$
Width after Phase 3	0
Width after Phase 4	$1 + \frac{1}{2}(\delta_{i,A_{11}} + \delta_{i,A_{21}})$
Redundant Wires	$(\delta_{i,A_{14}} + \delta_{i,A_{24}})$
Total Width	$2 + \frac{1}{2}(\delta_{i,A_{11}} + \delta_{i,A_{12}} + \delta_{i,A_{13}} + \delta_{i,A_{21}} + \delta_{i,A_{22}} + \delta_{i,A_{23}}) \leq k+2$
	Edge on Boundary Between A_1 and A_4
Width after Phase 1	0
Width after Phase 2	$1 + \frac{1}{2}(\delta_{i,A_{12}} + \delta_{i,A_{13}} + \delta_{i,A_{14}} + \delta_{i,A_{42}} + \delta_{i,A_{43}} + \delta_{i,A_{44}})$
Width after Phase 3	$1 + \frac{1}{2}(\delta_{i,A_{45}} + \delta_{i,A_{15}})$
Width after Phase 4	0
Redundant Wires	$(\delta_{i,A_{12}}\delta_{i,A_{42}})$
Total Width	$2 + \frac{1}{2}(\delta_{i,A_{13}} + \delta_{i,A_{14}} + \delta_{i,A_{43}} + \delta_{i,A_{44}}) \leq k+2$
	Edge on Boundary Between A_1 and B_1
Width after Phase 1	$k + 1 - \frac{1}{2}(\delta_{i,A_{11}} + \delta_{i,A_{12}} + \delta_{i,A_{13}} + \delta_{i,A_{14}} + \delta_{i,A_{15}} + \delta_{i,B_{11}} + \delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}} + \delta_{i,B_{15}})$
Width after Phase 2	0
Width after Phase 3	$1 + \frac{1}{2}(\delta_{i,A_{15}} + \delta_{i,B_{15}})$
Width after Phase 4	0
Redundant Wires	$(\delta_{i,A_{11}} + \delta_{i,A_{12}} + \delta_{i,A_{13}} + \delta_{i,A_{14}} + \delta_{i,B_{11}} + \delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}})$
Total Width	$\leq k+1$
	Edge on Boundary Between B_1 and C_l
Width after Phase 1	$k + 1 - \frac{1}{2}(\delta_{i,B_{11}} + \delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}} + \delta_{i,B_{15}} + \delta_{i,C_{11}} + \delta_{i,C_{12}} + \delta_{i,C_{13}} + \delta_{i,C_{14}} + \delta_{i,C_{15}})$
Width after Phase 2	0
Width after Phase 3	0
Width after Phase 4	$1 + \frac{1}{2}(\delta_{i,B_{11}} + \delta_{i,C_{11}})$
Redundant Wires	$(\delta_{i,B_{12}} + \delta_{i,B_{13}} + \delta_{i,B_{14}} + \delta_{i,B_{15}} + \delta_{i,C_{12}} + \delta_{i,C_{13}} + \delta_{i,C_{14}} + \delta_{i,C_{15}})$
Total Width	$\leq k+2$

From the preceeding discussions, it is clear that the channel width is at most $\frac{3}{2}k+4$.

As mentioned earlier, $(\frac{3}{2}k + 4)$ is the number of horizontal (vertical) wires over k consecutive nodes in a row (column). We can show that the result is tight as it is possible to construct an instance such that a set of k consecutive nodes in a row (column) would have at least $\lceil \frac{3}{2}k \rceil - 2$ wires passing over them. The following construction shows how to get at least $\lceil \frac{3}{2}k \rceil - 2$ wires over k consecutive nodes in a row / column .

First connect nodes $\{(i,j),(i,j+1),\cdots,(i,j+k-1)\}$ to the nodes $\{(i,j+k),(i,j+k+1),\cdots,(i,j+2k-1)\}$ respectively. Then connect nodes $\{(i+1,j+2k-1),(i+1,j+2k),\cdots,(i+1,j+3k-2)\}$ to the nodes $\{(i+1,j+3k-1),(i+1,j+3k),\cdots,(i+1,j+4k-2)\}$ respectively. Clearly, there are k horizontal wires between nodes $(i,j+k-1)$ and $(i,j+2k-2)$, and the node $(i+1,j+2k)$ and $(i+1,j+3k)$. Now define $k-2$ nets by the following nodes : $[(i+1,j+2k-2),(i,j+3k-3)],[(i+1,j+2k-3),(i,j+3k-4)],\cdots,[(i+1,j+k+1),(i,j+2k)]$

Nets $[(i+1,j+2k-2),(i,j+3k-3)]$ and $[(i+1,j+k+1),(i,j+2k)]$ can be routed so that they do not go over the k nodes between the nodes $(i,j+k-1)$ and $(i,j+2k-2)$, and the node $(i+1,j+2k)$ and $(i+1,j+3k)$. This is illustrated in Figure 10. Thus upon routing, the sections between the nodes $(i,j+k-1)$ and $(i,j+2k-2)$, and the node $(i+1,j+2k)$ and $(i+1,j+3k)$ would have at least $(k+\lceil \frac{k-4}{2} \rceil)$ horizontal wires. Similar construction holds for the k consecutive nodes in the column.

5. Time Complexity of Routing :

From theorem 2 , we know that the integral approximation is obtained in time $O(r^3 log(1+s/r)+r^3+d^2r+sr)$, when it is known that $Ax=b$ and A is a $(r \times s)$ real-valued matrix, x is s vector and b is r vector and x contains d distinct components. In the setup for routing of a $2k \times 2k$ subgrid, $r=4k$, $s=(2k^2-\frac{y}{2})$ where $0 \leq y \leq 4k^2$ and $d=$ *constant* . Thus each subgrid is routed in time $O(k^3 \ log \ k)$. Also during each phase, at most $O(\frac{N^2}{k^2})$ subgrids are routed. Thus, the overall time complexity is :

$$\frac{N^2}{k^2} O(k^3 \ log \ k)$$

$$\text{or } O(N^2 k \ log \ k)$$

As a result of the above discussions, we have following theorem :

Theorem 3

$$\lceil \tfrac{3}{2}k \rceil - 2 \leq W_1^k(k) \leq \lceil \tfrac{3}{2}k \rceil + 4$$

The routing is performed in time $O(N^2 k \ log \ k)$.

6. Parallel Implementation

The routing in any one $2k \times 2k$ subgrid during any of the four phases is completely independent of the routing in any other $2k \times 2k$ subgrid during the same phase. Thus, the same routing technique could be implemented in parallel on $O(\frac{N^2}{k^2})$ processors in $O(k^3 \ log \ k)$ parallel time. Since all the wires are $O(log^{\frac{3}{4}} N)$, we have a polylogarithmic time wafer routing algorithm.

7. Conclusion

In this research we have developed efficient techniques for routing two terminal nets on an $N \times N$ grid under the restriction that each net is of length at most k, where k is a small constant or a small function of N. The routing techniques outperform the previously stated methods by Karp et. al. and Raghavan and Thompson. Our results are deterministic.

8. References

[1] Bhatia D., Leighton T., Makedon F., Efficient Reconfiguration of WSI Arrays, *Proceedings of first International Conference on System Integration, IEEE CS Press, pp. 47-56.*

[2] Burstein M., Pelavin R., Hierarchical Wire Routing, *IEEE Transaction on CAD, Vol. CAD-2, No. 4, pp. 223-234, October 1983*

[3] Karp R. M., Leighton F.T., Rivest R.L., Thompson C. D., Vazirani U.V., Vazirani V.V., Global Wire Routing in Two Dimensional Arrays, *Algorithmica, 1987, 2:113-129, Springer Verlag N.Y. Inc.*

[4] Leighton T., Leiserson C., A Survey of Algorithms for Integrating Wafer Scale Systolic Arrays, *Proceedings of workshop on WSI IFIP WG 10.5, Grenbole, France .*

[5] Raghavan P., Thompson C.D., Provably Good Routing in Graphs : Regular Arrays, *Proceedings of the 17th STOC, ACM, New York, 1985, 79-87*

[6] Vecchi M.P., Kirkpatrick, Global Wiring by Simulated Annealing, *IEEE Transaction on CAD, Vol. CAD-2, No. 4, pp. 215-222, October 1983*

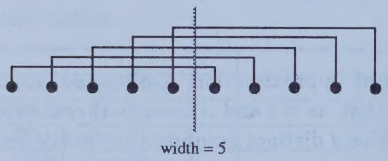


Figure 1

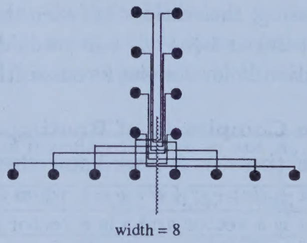


Figure 2

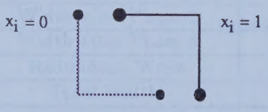


Figure 3

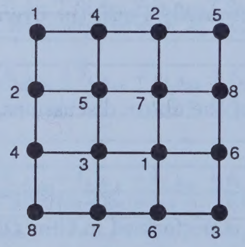


Figure 4

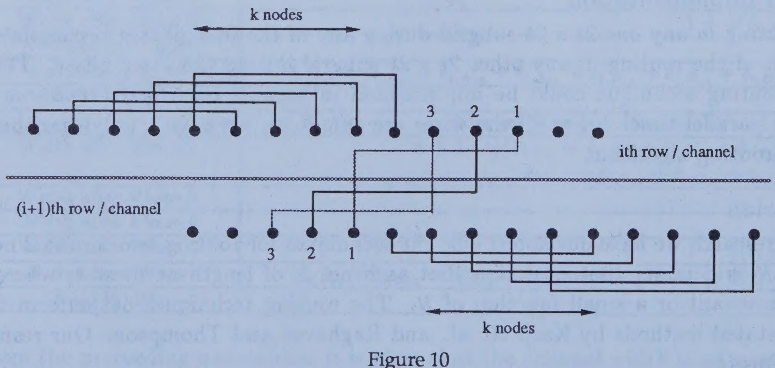


Figure 10

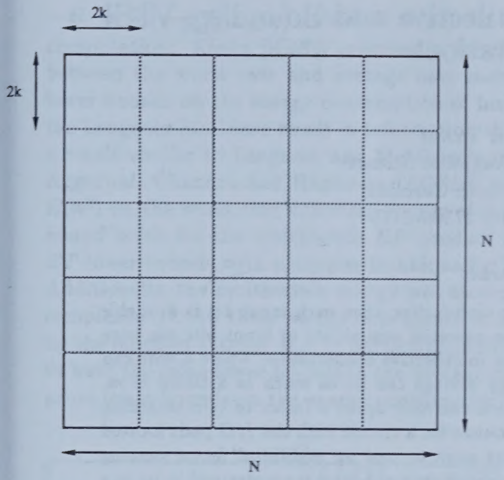


Figure 5

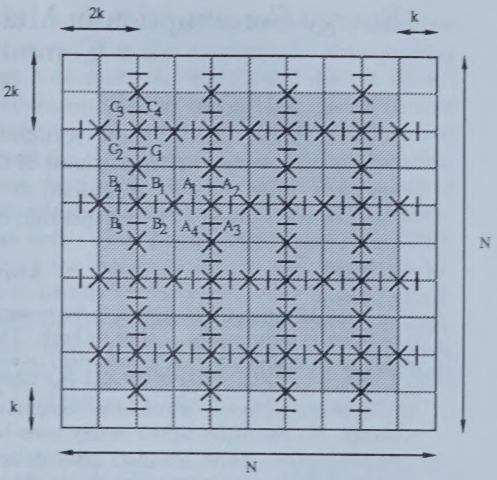

2k
k
2k
C3
C4
C2
C1
B4
B1
A1
A2
B3
B2
A4
A3
k
N
N

Figure 6

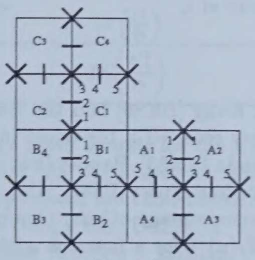

C3
C4
3 4 5
2
1
C2
C1
B4
B1
1
2
3 4 5
A1
A2
1 1
2 2
5 4 3 3 4 5
B3
B2
A4
A3

Figure 7

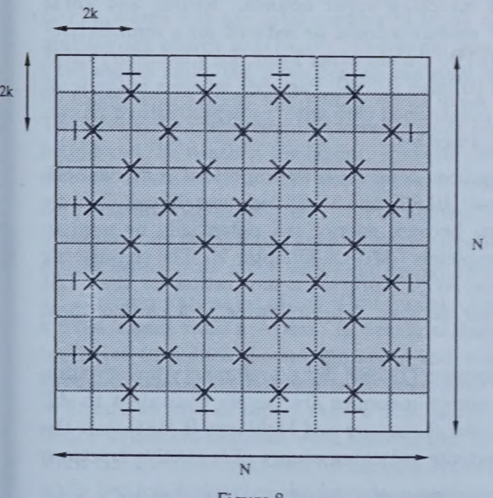


Figure 8

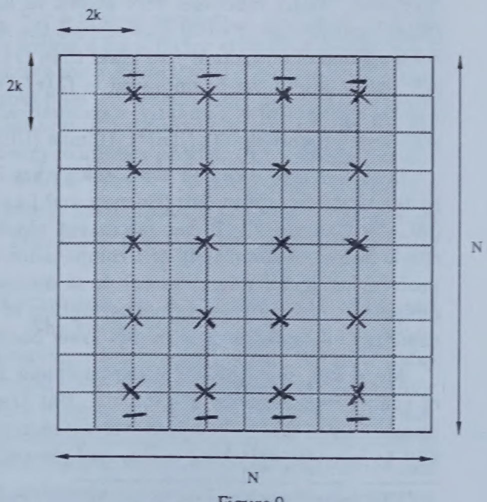


Figure 9

Energy Consumption in Multilective and Boundary VLSI Computations*

AKHILESH TYAGI
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175

Abstract

A VLSI computation is said to be m-way *multilective* when each input bit is available m times in either space or time or both. The repeated availability of input bits can save computational energy. For a uniswitch, m-way multilective computation, where a wire can switch at most once, we show that the energy savings can be as much as a factor of m. A multiswitch, m-way multilective computation can save up to a factor of $\sqrt{m}$ switching energy. We also derive tighter energy lower bounds for a circuit with the I/O pads located on the border. These boundary computations seem to cost an additional factor ranging from $\sqrt{\log n}$ to $\log n$ in switching energy. The circuits whose layout is constrained to be in a region with aspect ratio a also require additional energy. The additional energy cost ranges from a factor of $\sqrt{a}$ to a factor of a.

1 Introduction

Thompson [Tho79] and Brent, Kung [BK80] laid the foundation for VLSI complexity theory. The area-time complexity theory received a big boost in the early 80s with many researchers contributing to the field: Vuillemin [Vui83], Baudet [Bau81], Lipton and Sedgewick [LS81], Yao [[Yao79], [Yao81]], Papadimitriou and Sipser [PS82], Aho, Ullman and Yannakakis [AUY83], and Bilardi and Preparata [BP86] among many others. One main result established in [Tho79] and [BK80] states that AT^2 is $\Omega\left(I_f(n)^2\right)$ for a function with information complexity I_f. Here, A and T are the area and time, respectively, of a VLSI chip computing f. The latter research discovered many new and powerful techniques to bound $I_f(n)$ from below for a large class of functions. Many functions were shown to have matching upper bounds. Kedem and Zorat [KZ81] and Kedem [Ked82] showed that the AT^2 resources could be reduced for a *multilective*[†] computation of a function f to lower than $\Omega\left(I_f(n)^2\right)$. On the other hand, for a boundary chip[‡] AT^2 lower bound can be improved to $\Omega\left(rI_f(n)\right)$ [Ullman [Ull84], Page 58], where r is the total number of input bits. Similarly, a geometrically constrained chip with an aspect ratio a has an AT^2 lower bound of $\Omega\left(aI_f(n)^2\right)$ [Ullman [Ull84], Page 57].

In addition to *area* and *time*, *energy* has long been considered an important resource both by the theoretical physicists [Bennett and Landauer [BL85]] and VLSI designers [Mead, Conway [MC80], Chapter 9]. It has fascinated physicists because energy can potentially be used to characterize the complexity of a computation completely [[MC80], Chapter 9]. The engineering problems of metal migration and heat dissipation [WE85] provide a motivation for the VLSI designers to monitor energy consumption of their designs. The following is a chronological summary of the research in energy lower bounds.

Mead and Rem [[MC80], Chapter 8] and Thompson [Tho80] did a preliminary investigation of energy requirements in a circuit. But their energy measures are largely equivalent to AT measure since they only consider static energy model. Lengauer and Mehlhorn [LM81] were the first to consider switching energy and showed that the switching energy of transitive functions

---

[*] This research was supported in part by NSF Grant #MIP-8806169, NCBS&T Grant #89SE04 and a Junior Faculty Development award from UNC, Chapel Hill.

[†] A multilective protocol allows an input bit to be available more than once either in space or in time.

[‡] Where all the input/output pads are constrained to be on the boundary of the chip.

– E, is $\Omega\left(\frac{n^2}{P\log(AP^2/n^2)}\right)$, which is $\Omega(n^2)$ for $AP^2 = O(n^2)$. P is the period of a pipelined computation. Kissin [Kis82] proposed a formal model of energy consumption distinguishing between the worst case and average case energy consumptions. She [Kis85] derived several lower bounds on the energy consumption of functions such as *and* and *or*. Leo [Leo84] derived the Lengauer-Mehlhorn result in information-theoretic terms. Snyder and Tyagi [ST86] derived a result similar to Lengauer and Mehlhorn's result for the average case energy consumption. Aggarwal, Chandra and Raghavan [ACR88] were the first to obtain a tight lower bound of $\Omega(n^2)$ on the worst case uniswitch energy of shifting using a novel cutting method. This lower bound holds for the multiswitch EP product as well. Tyagi [Tya89] rederived the [ACR88] ET lower bounds with a simpler technique for both the uniswitch and the multiswitch model. Additionally, the multiswitch energy was shown to be $\Omega(I^{3/2})$ for a function with information complexity I[§]. All these results were shown to hold for the average case energy with I replaced by the function entropy H_f. In particular, shifting, integer multiplication and DFT were shown to have the same lower bounds in the average case as in the worst case. Tyagi and Reif [TR90] prove lower bounds on the energy consumption in 3-dimensional and optical computations.

	USM energy, USM and MSM ET product	*MSM energy*
Worst-case lower bound	$\Omega\left(m\left(I(m)\right)^2\right)$ or $\Omega\left(\frac{I^2}{m}\right)$	$\Omega\left(m\left(I(m)\right)^{3/2}\right)$ or $\Omega\left(\frac{I^{3/2}}{\sqrt{m}}\right)$
Average-case lower bound	$\Omega\left(\frac{H_f^2}{m}\right)$	$\Omega\left(\frac{H_f^{3/2}}{\sqrt{m}}\right)$

In this paper, we extend the energy lower bounds for the multilective case, for a boundary chip and for a chip with aspect ratio a. In particular, we show the following lower bounds. For a multilective chip with m-way repetition, the table above states our lower bounds, where $I(m)$ refers to the m-way information complexity [Definition 1, Section 3]. All the cases have a matching upper bound for a transitive function – shifting. *This establishes that the input repetition can save energy.*

For a chip with I/O pads on the border, ET and uniswitch energy have a lower bound of $\Omega(rI)$, where r is the total number of input bits. For a typical problem, r is $\Omega(n\log n)$. The multiswitch energy is $\Omega\left(n\sqrt{r}\right)$. The upper bounds within a $\log n$ factor exist for shifting. For a fixed aspect ratio, the lower bounds can only be shown for a *dense* layout. We argue why, from an energy stand-point, our definition of a dense layout does not preclude any interesting circuits. This was not an issue for AT^2 lower bounds because area of a convex chip only depends on the height and the width of the layout. However the energy exhibits a peculiar mix of characteristics of both area and time. We show a lower bound of $\Omega(aI^2)$ on ET and uniswitch energy of a dense layout with aspect ratio a. The multiswitch energy lower bound is $\Omega\left(\sqrt{a}I^{3/2}\right)$. Matching upper bounds for shifting are also demonstrated. *This shows that a boundary computation and a fixed aspect ratio incur additional energy cost.*

All three variations of the VLSI model – multilective protocol, I/O pads on the border and a fixed aspect ratio a – occur in practical designs. The replication of input bits is a common technique to reduce the storage needs and wire lengths. Almost all the chips designed today have the pads on the border due to the ease of bonding. And having to design a circuit module to fit into an area of aspect ratio a is all too common a scenario in automated layout synthesis. Hence we believe that this work adds further to our understanding of the energy consumption in VLSI circuits. The next section introduces the model. Section 3 deals with the multilective case while Section 4 sketches the boundary and aspect ratio constrained computations. The upper bounds are given in Section 5 and Section 6 sketches the extensions to the average energy case.

[§] For the sake of brevity, we will use I to denote $I_f(n)$

2 Model

The model of VLSI computation is essentially the same as the one described by Thompson [Tho79]. A computation is abstracted as a communication graph. A communication graph is very much like a flow graph with the primitives being some basic operators that are realizable as electrical devices. Two communicating nodes are adjacent in this graph. A layout can be viewed as a convex embedding of the communication graph in a Cartesian grid. Each grid point can either have a processor or a wire passing through. A wire cannot go through a grid point with a processor unless it is a terminal of the processor at that grid point. The number of layers is limited to some constant γ. Thus both the fanin and fanout are bounded by 4γ. Wires have unit width and bandwidth and processors have unit area. The initial data values are localized to some constant area, to preclude an encoding of the results. The input words are read at the designated nodes called input ports. The input is synchronous and each input bit is available only once except for the multilective protocol. The multilective protocol allows an input bit to be replicated either in space or in time or both. The input and output conventions are where-determinate but need not be when-determinate.

The following is an enhancement of this model to account for energy. It is more or less similar to the model proposed by Kissin [Kis82], [Kis85] except for the multiswitch energy case. We assume that whenever a wire of length l changes state, it consumes $\Theta(l)$ switching energy. We do not account for the switching energy consumed by the processors. We distinguish between the following two cases. When a wire can switch at most once during a computation, it is a Uniswitch computation. It was named Uniswitch Model (USM) by Kissin [Kis82]. In contrast, if a wire is allowed to switch any number of times, we get the Multiswitch model (MSM). We denote a circuit $C(V, W)$ as a graph, where V is the set of nodes and W is the set of edges or wires in $V \times V$. Notice that a circuit is allowed to be cyclic.

We define a measure of energy consumption for a circuit. When a circuit C is subjected to an input $\vec{x}$, let wire w_i switch k_i times before the circuit is settled. Let l_i be the length of the wire w_i in the circuit C's embedding in a grid. Then the energy consumption for circuit C, $E_W(C, s, \vec{x})$ in state s with $\vec{x}$ as the input is defined to be $u \times \sum_{w_i \in W} k_i \times l_i$. For the Uniswitch model $k_i \leq 1$. We will distinguish between worst case energy consumption and average case energy consumption. The worst case energy consumption for a circuit C, $E_w(C)$, is $max_{s,\vec{x}} E_W(C, s, \vec{x})$, where the maximum is taken over all (state, input vector) pairs. The average case energy consumption for a circuit C is its energy consumption averaged over all initial states and all input vectors.

For further details of this model, the reader is referred to either of the theses [Kis87], [Tya88] or to Kissin's papers [Kis82], [Kis85].

3 Multilective Energy Consumption

In this section, we consider the uniswitch and multiswitch energy lower bounds for the multilective case. Let us consider m-way multilective model – *every input bit is repeated m times.* We choose this definition of m-way multilective computation for the convenience of illustration. The alternative way, used by Kedem and Zorat [KZ81], states that the sum of occurrences of each input bit, l_i, $\sum_i l_i$ be mn. The results do not change with this alternative definition. Let us first define the notion of m-way information complexity, which differs from the m-way fooling set of Ullman [[Ull84], Page 70].

Definition 1 *Let $Z = (X_1, X_2, \ldots, X_m)$ be an overlapping partition of the set of input variables with $(Y_1, Y_2, \ldots, Y_m)$ being the corresponding partition of the output variables. Note that X_is are multisets. In a* valid partition, *each input variable x_j has exactly m occurrences in $\cup_{i=1}^{m} X_i$ and the number of input bit occurrences in X_i, $|X_i|$, for all $1 \leq i \leq m$ is at most a constant fraction of n, say $\frac{2n}{3}$. Let $I^Z_{i,j}(f, x)$ be the number of bits that must be exchanged between X_i and X_j with* *an input assignment x for a correct computation of f. Then the m-way information complexity of f with respect to an m-way valid partition Z, $I_f^Z(m)$, is defined to be $\max_x \min_{i,j} I_{i,j}^Z(f,x)$. The minimum is taken over all the pairs $(X_i,\ X_j)$ in the partition with the maximum taken over all the input assignments. The m-way information complexity, $I_f(m)$, is $\min_Z I_f^Z(m)$ where the minimum is taken over all the valid m-way partitions Z.*

Kedem and Zorat [KZ81] and Ullman [Ull84] define $I(m)$ as the maximum number of bits traversing into some X_k during the course of computation. Ullman shows that for shifting $I(m)$ is $\Omega\left(\frac{n}{m}\right)$. Our definition of m-way information is weaker. However, Corollary 1 gives a lower bound on energy and energy-time product in terms of the information complexity I and m, $\Omega(I^2/m)$. Even if we had been able to use Kedem, Zorat and Ullman definition of m-way information, this is the best lower bound we could have derived. As we show in Section 5, shifting has a matching upper bound.

The lower bound proof requires a new cutting technique. Neither the energy lower bound partitions of Aggarwal *et al.* [ACR88] and Tyagi [Tya89] nor the multilective cut of Kedem, Zorat [KZ81] work in this situation. In the following, our intent is to cut the chip in such a way as to derive at least $m/17$ regions with the following characteristics. Each region contains at least $n/20$ and at most $17n$ input bit occurrences[¶]. Each region has an internal seed – a rectangle containing at least $n/20$ and at most $n/5$ input bit occurrences as shown in Figure 1. Let the internal rectangle for a region have perimeter p. Then $p/4$ rings around the perimeter of the seed rectangle are contained completely inside the region. Under these conditions, energy consumption can be summed up over these disjoint regions.

We use a *square tessellation*, as defined in Bilardi, Preparata [BP86], as an initial cut. Let the chip be oriented in the Cartesian coordinates with the bottom-left corner aligned with the origin. A square tessellation with spacing l creates square-shaped cells with the bottom-left and top-right corner coordinates $(il,\ jl)$ and $([i+1]l,\ [j+1]l)$ respectively for any two integers $i, j \geq 0$. We also need the notion of a multilective minimum bisection width $\delta(s)$ which is different from the minimum bisection width used in [Tho79] and [BP86]. Unlike the definition of the minimum bisection width in Thompson [Tho79], we wish to cut a certain number of wires and place an upper bound on the number of input bit occurrences thus isolated. As defined in Bilardi, Preparata, a *dichotomy* of a circuit $C = (V, W, \mathcal{I}, \mathcal{O})$ is a partition $(V_1,\ V_2)$ of V and $\delta(V_1, V_2)$, *the dichotomy width*, is the number of edges connecting V_1 and V_2. Let $\mathcal{I}_m$ be the set of mn input bit occurrences in an m-way multilective computation, $x_1^0,\ x_1^1,\ \ldots,\ x_1^{m-1},\ x_2^0,\ \ldots,\ x_n^{m-1}$. Now let us define the *multilective minimum bisection width* $\delta(s)$.

Definition 2 *Given a circuit $C = (V, W, \mathcal{I}, \mathcal{O})$ and an integer $0 < s \leq |\mathcal{I}_m|$, let Π_s be the set of all partitions $\{(V_1,\ V_2)$ s.t. $|V_1 \cap \mathcal{I}_m| = s\}$. Then the multilective minimum bisection width $\delta(s)$ is defined as $\min_{(V_1,\,V_2)\in\Pi_s}\delta(V_1,\ V_2)$.*

In other words, $\delta(s)$ is the minimum number of wires (edges) in the computation graph $C = (V, W)$ that must be removed to separate a region containing at most s input bit occurrences. Let us state and prove the cutting lemma.

Lemma 1 *Given a chip $C = (V, W)$, there exist at least $m/17$ disjoint regions with the following property. Each region contains at least $n/20$ and at most $17n$ input bit occurrences. There exists a rectangle R inside each region containing at least $n/20$ and at most $n/5$ input bit occurrences. Let the perimeter of the R be p. Then the horizontal and vertical distance from the perimeter of R to the boundary of the region is at least $p/4$.*

[¶] Each of the m occurrences of an input bit counts as an *input bit occurrence* for a total of mn input bit occurrences.

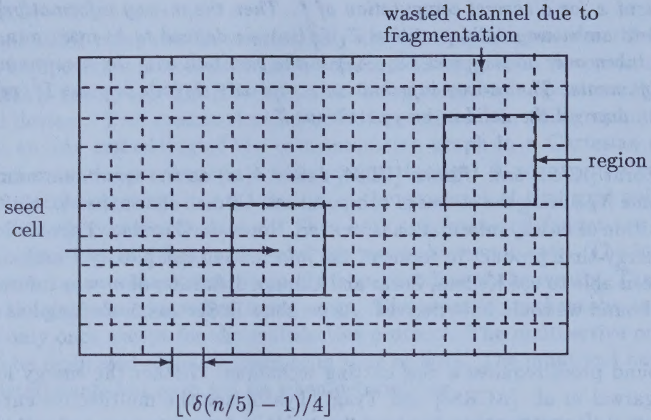

wasted channel due to fragmentation
region
seed cell
$\lfloor(\delta(n/5) - 1)/4\rfloor$

Figure 1: An Example of the Tessellation for Multilective Cut

PROOF: The proof uses the idea of square tessellation from Bilardi and Preparata [BP86]. The intention is to get reasonably square shaped regions with the desired properties. Consider a square tessellation [BP86] of the chip with spacing $l = \lfloor(\delta(n/5) - 1)/4\rfloor$. Each cell in the tessellation is a $l \times l$ square. We claim that each cell contains at most $n/5$ input bit occurrences. The proof follows the reasoning in Theorem 1 of [BP86]. If a cell contains more than $n/5$ input bit occurrences, we can divide it into two parts using a zig-zig cut [as in [Tho80], page 55] such that one part contains exactly $n/5$ input bit occurrences. The perimeter of this part is bounded by $4l$, which is at most $\delta(n/5) - 1$. Hence it cannot contain $n/5$ input bit occurrences as it violates Definition 2. Most of the cells will be marked off as a region in one of the several phases $t = 0, 1, \ldots$ through the following procedure.

Let us consider the tessellation during the tth phase. The cells have the perimeter $4\,2^t\,l$ and contain at most $n/5$ input bit occurrences. Consider the cells that contain at least $n/20$ input bit occurrences. These cells form the *seed* rectangles. Such a cell along with eight adjoining cells constitutes a region. In case of conflicts between two neighboring cells, the cell with a higher input bit occurrence count in its surrounding region wins. If any of the surrounding cells of a candidate seed cell is already marked off from one of the previous phases, a new region around this seed cell cannot be formed. All the cells included in a region are marked off and the tessellation spacing is doubled for the $(t+1)$st phase.

The following invariant holds during any of the tessellation phases. Any cell can contain at most $n/5$ input bit occurrences because it is the union of 4 lower level (tth phase) unmarked cells containing less than $n/20$ input bit occurrences. This process stops when every cell is either marked off or if all the cells contain less than $n/20$ input bit occurrences and they cannot be extended any further.

How many regions are formed by this procedure? We claim that the number of input bit occurrences contained in a region or lost in fragmentation due to this region [contained in an unmarked cell] is at most $17n$. The number of input bit occurrences in a region is at most $n/5 + 8\frac{n}{5}$, for the seed cell and eight surrounding cells. Let us consider the input bit occurrences lost in the fragmentation of the chip by this procedure. All the unmarked cells at the end of this procedure did not participate in a region for one of the two reasons. Either it was a candidate in one of the earlier phases, but some of its surrounding cells were already marked or it never became a candidate seed cell. All the unmarked cells can be associated with a marked region. Even a generous counting of the number of input bit occurrences in the unmarked cells associated with a marked region gives us an upper bound of $17n$ as follows. There can be at most eight of the first type of unmarked cells (which were a candidate seed cell) associated with a marked region. In such a case, at most $8\frac{9n}{5}$ input bit occurrences per region are lost to the fragmentation. The number of input bit occurrences in all of the low-density channels (the second type of unmarked cells) around a marked region is bounded by $8\frac{n}{20}$. In either case, the maximum number of input bit occurrences in unmarked cells attributable to one marked region along with the number of input bit occurrences in a marked region does not exceed $\frac{81n}{5} < 17n$. Since each of the mn occurrences of input bits is contained in one of the marked regions or its associated unmarked cells, there must be at least $m/17$ regions.

□

Let us first show the energy-time product lower bounds for both uniswitch and multiswitch cases. We will use the following information-theoretic lemma proved in Tyagi [Tya88], [Tya89]. The lemma asserts that to transmit k information bits using $k' > k$ bits, at least $\Omega\left(\frac{k}{\log(k'/k)}\right)$ switching must be incurred.

Lemma 2 *Let k' and k be two positive integers such that $k' \geq k$. The average number of alternations in transmitting a k' bit encoding of k information bits, $A(k', k) = \Omega(k/4\log[(k'/k)+2])$.*

Theorem 1 *For a multilective chip with m-way repetition, both multiswitch and uniswitch ET product and uniswitch energy have a lower bound of $\Omega\left(m\left(I(m)\right)^2\right)$, where $I(m)$ is the m-way information complexity of the function f being computed.*

PROOF SKETCH: Consider the disjoint region partition provided by Lemma 1. Let us concentrate on a single region. Consider its seed rectangle R with a perimeter p. Let $W(0)$ be the set of all the horizontal or vertical unit length wire segments crossing the perimeter of R. Let $W(i)$ be defined recursively as the set of unit length wires adjacent to the wires in $W(i-1)$ and not in $W(0), \ldots, W(i-1)$ and not inside R. These are the rings around R. Let us consider the rings $W(0), W(1), \ldots W((p/4)-1)$. Note that $W(0)$ contains at most $2p$ wires. It can also be shown that the number of wires in $W((p/4)-1)$, $|W((p/4)-1)| \leq 16p$. Consider the switching energy at a ring $W(i)$. The total number of bits transmitted across a ring, q, cannot exceed $16pT$. If the majority of the output bits are generated inside a ring $W(i)$, then the information about the input bits read in the rest of the chip needs to come into the ring. Otherwise, the information about the input bits read inside the ring needs to go out. In either case, since R contains at least $n/20$ and at most $n/5$ input bit occurrences, by Definition 1 at least $I(m)$ information bits must cross the ring $W(i)$. Hence the energy consumption at a ring $W(i)$ is $\Omega\left(\frac{I(m)}{\log(16pT/I(m))}\right)$ by Lemma 2. Summed up over $p/4$ rings, the energy E is $\Omega\left(\frac{I(m)\,p}{\log(pT/I(m))}\right)$. Summing this energy over $m/8$ disjoint regions, the total energy is $\Omega\left(\frac{m\,I(m)\,p}{\log(pT/I(m))}\right)$. This gives an ET product lower bound of $\Omega\left(m\,\frac{I(m)\,pT}{\log(pT/I(m))}\right)$. Note that the actual number of bits transmitted $16pT \geq q \geq I(m)$. The expression $\frac{pT}{\log(pT/I(m))}$ attains its minimum value at $pT = I(m)$ giving an ET lower bound of $\Omega\left(m\left(I(m)\right)^2\right)$. For uniswitch energy lower bound, note that since every wire can carry at most one bit of information during the course of entire computation, the maximum number of bits transmitted across a ring is $16p$ rather than $16pT$. Hence the energy across a ring is $\Omega\left(\frac{I(m)}{\log(16p/I(m))}\right)$. Also notice that $p \geq I(m)$ and hence energy per region is $\Omega\left(I(m)^2\right)$. Over $m/17$ regions, the energy is $\Omega\left(m\,I(m)^2\right)$.

□

Theorem 2 *The multiswitch energy of a multilective computation is* $\Omega\left(m\,(I(m))^{3/2}\right)$.

Sometimes, it is not a trivial task to derive lower bounds on the m-way information complexity of a function f. However the information complexity of many functions is already known in the literature. Hence a characterization of these lower bounds in terms of I and m is useful. The following corollary does that.

Corollary 1 *For a multilective chip with m-way repetition, both multiswitch and uniswitch ET product and uniswitch energy have a lower bound of* $\Omega\left(\frac{I^2}{m}\right)$, *where I is the information complexity of the function f being computed. The multiswitch energy of a multilective computation is* $\Omega\left(\frac{I^{3/2}}{\sqrt{m}}\right)$.

4 Boundary and Fixed Aspect Ratio Computations

Let us consider a chip with the input/output pads on the boundary. It is assumed that $r \geq n$ input bits are read through these input pads located at the convex border. For a typical problem, $r \geq n \log n$ gives us a stronger lower bound for this special case than the general lower bound of $\Omega(n^2)$.

Theorem 3 *For a boundary chip reading r input bits, the ET product and uniswitch energy are* $\Omega(r\,I)$ *for the computation of a function with information complexity I. The multiswitch energy is* $\Omega\left(\sqrt{r}\,I\right)$.

PROOF SKETCH: Without loss of generality we assume that the pads are located on a rectangular boundary. At least one side of this rectangle reads at least $r/4$ input bits. Consider a chord perpendicular to this side such that at least $r/12$ input bits are read on either side of this chord. This assumes that no port reads more than $r/12$ input bits, otherwise the proof is trivial. Let $q \geq I$ bits be transmitted across this cut C. The remaining proof is very much along the lines of the proof of Theorem 2. In the uniswitch case, the average distance from an input node to a unit segment wire in C is $\Omega(r)$. From Lemma 2, at least $\Omega(I/\log(q/I))$ paths switch giving an energy bound of $\Omega\left(\frac{I(q+r)}{\log(q/I)}\right)$. For $q \geq I$, this minimizes at $\Omega(r\,I)$. For the multiswitch case, using a variant of Baudet's result [Bau81] and Paterson, Ruzzo, Snyder result [PRS81], the average distance from an input node to a cut wire is $\Omega\left(\sqrt{q+r}\right)$. This gives a multiswitch energy lower bound of $\Omega\left(\sqrt{r}\,I\right)$.

□

Now let us consider the fixed aspect ratio lower bounds. Our lower bounds apply only to *dense* layouts. In an intuitive sense, a $h \times l$ layout is dense if the larger fraction of the area is used gainfully. Consider the shifter shown in Figure 2. It has an aspect ratio of $\sqrt{n}$. Although, only a single wire is stretched to the length n. The energy consumption of this shifter is still $O\left(n^{3/2}\right)$. The actual area is still $O(n)$ but the convex area is $n^{3/2}$. We wish to preclude such designs from consideration as dense layouts. The weakest condition that precludes such layouts seems to say "given a region containing a good fraction of input bits, say $n/81$, the cut separating it from another region with area that is a significant fraction of the total chip area, say $A/16$, carries $\Omega(I_f(n))$ information during the course of computation".

Theorem 4 *For a dense layout, a VLSI computation of a function f constrained in an aspect ratio a has a lower bound of* $\Omega(a\,I^2)$ *on the uniswitch and multiswitch ET product and uniswitch energy. The multiswitch energy lower bound is* $\Omega(\sqrt{a}\,I^{3/2})$.

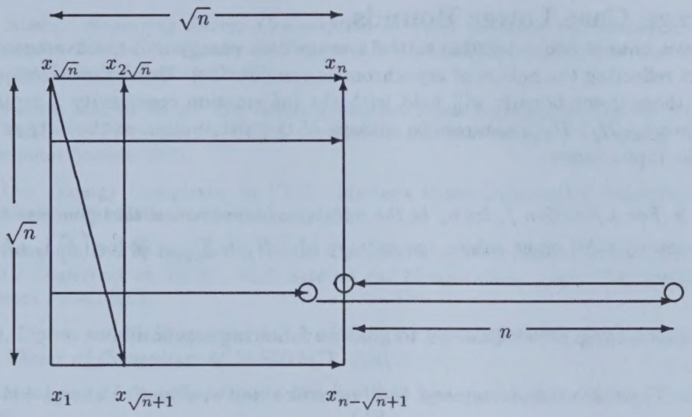


Figure 2: A Non-Dense Shifter with Aspect Ratio Square Root n

5 Upper Bounds

We will show all the upper bounds on a transitive function – shifting. Note that the m-way information complexity of shifting is n/m and its information complexity is given by $\Omega(n)$. All the upper bounds are given by the following catalog of shifter designs. The barrel shifter is the well-known butterfly type of network with area $O(n^2)$, time $O(\log n)$ and uniswitch energy $O(n^2)$. The square shifter is a $\sqrt{n} \times \sqrt{n}$ shifter described in Ullman [[Ull84], Page 69]. It has area $O(n)$, time $O(\sqrt{n})$ and multiswitch energy $O(n^{3/2})$. The square shifter can be used with repeated input bits with m-way repetition as described in Ullman [[Ull84], Page 70]. For instance, for $m = n^{1/3}$, the area is $O(n)$, time $O(n^{1/3})$ and multiswitch energy is $n^{4/3}$. The m-way repetition for a uniswitch shifter is demonstrated in Kedem, Zorat [KZ81]. This shifter has area $O(nm + n^2/m)$, time $O(\log m + \log(n/m))$ and uniswitch energy equal to A.

Thus the m-way uniswitch ET product of a Kedem-Zorat barrel shifter for $m \leq \sqrt{n}$ is $O\left(\frac{n^2 \log(n/m)}{m}\right)$. The lower bound is $\Omega(n^2/m)$. The uniswitch energy is $O(n^2/m)$ which is the lower bound. The m-way square shifter for $m = n^{1/3}$ has multiswitch ET of $O(n^{5/3})$ and the multiswitch energy of $O(n^{4/3})$. The corresponding lower bounds are $\Omega(n^{5/3})$ and $\Omega(n^{4/3})$ respectively.

For a boundary chip, note that a barrel shifter is a boundary chip and so is a square shifter if all the four sides read all the input bits and then propagate them to the inside plane in time $O(\sqrt{n})$. To make it interesting, let us shift $\log n$ bit words. Then the barrel shifter has area $O(n^2 \log^2 n)$, time $O(\log n)$, and uniswitch energy $O(n^2 \log^2 n)$. The uniswitch lower bound on both ET and E is $\Omega(n^2 \log n)$. A $\log n$ word square shifter has area $O(n \log n)$, time $O(\sqrt{n})$ and multiswitch energy $O(n^{3/2} \log n)$. The multiswitch ET and E lower bounds are $\Omega(n^2 \log n)$ and $\Omega(\sqrt{\log n}\, n^{3/2})$ respectively.

The same shifter designs provide the upper bounds for the aspect ratio as well for $a = 1$ which is not very interesting. A linear shift register with n shifting elements connected in a chain has area $O(n)$, time $O(n)$ and multiswitch energy $O(n^2)$. The aspect ratio of this design is n. The lower bound on the multiswitch ET and E are $\Omega(n^3)$ and $\Omega(n^2)$ respectively. It does not seem possible to build a dense uniswitch shifter with a non-constant, $\omega(1)$, aspect ratio.

6 Average Case Lower Bounds

All these lower bounds are extensible to the average case energy and the average case energy-time product reflecting the nature of asynchronous computation. Using the techniques in Tyagi [Tya89], all these lower bounds will hold with the information complexity I replaced by the function entropy – H_f. H_f measures the entropy of the distribution of the output values with respect to the input values.

Definition 3 *For a function f, let n_y be the number of input values that give rise to the output value y. Assuming n-bit input values, the entropy of f H_f is $\sum_{y \in O} \frac{n_y}{2^n} \log\left(\frac{2^n}{n_y}\right)$, where O is the set of output values.*

The results of Tyagi [Tya89] extend to give the following corollaries.

Corollary 2 *The uniswitch energy and both uniswitch and multiswitch energy-time product of a $m-$way multilective computation are $\Omega\left(\frac{H_f^2}{m}\right)$ for a function f with function entropy H_f.*

Corollary 3 *The multiswitch energy of a m-way multilective computation is $\Omega\left(\frac{H_f^{3/2}}{\sqrt{m}}\right)$ for a function f with function entropy H_f.*

In Tyagi [Tya89], the function entropies of DFT, shifting and multiplication are shown to be $\Omega(n)$. This provides as good a lower bound on the average case energy of these functions as is known for the worst case energy. The asynchronous computation does not seem to hold any asymptotic advantage over synchronous computation for a multilective computation of DFT, shifting and integer multiplication.

References

[ACR88] A. Aggarwal, A. K. Chandra, and P. Raghavan. Energy Consumption in VLSI Circuits. In *Proceedings of ACM Symposium on Theory of Computing*, pages 205–216. ACM-SIGACT, 1988.

[AUY83] A. Aho, J.D. Ullman, and M. Yannakakis. On Notions of Information Transfer in VLSI Circuits. In *Proceedings of ACM Symposium on Theory of Computing.* ACM-SIGACT, 1983.

[Bau81] G. M. Baudet. On the Area Required by VLSI Circuits. In *Proceedings of CMU Conference on VLSI - VLSI Systems and Computations*, pages 100–107. CMU, Computer Science Press, 1981.

[BK80] R.P. Brent and H.T. Kung. The Chip Complexity of Binary Arithmetic. In *Proceedings of ACM Symposium on Theory of Computing.* ACM-SIGACT, 1980.

[BL85] R. Bennett and R. Landauer. Fundamental Physical Limits of Computation. *Scientific American*, pages 48–58, July 1985.

[BP86] G. Bilardi and Preparata F. P. Area-Time Lower-Bound Techniques with Applications to Sorting. *Algorithmica*, 1(1):65–91, 1986.

[Ked82] Z. Kedem. Optimal Allocation of Computational Resources in VLSI. In *Proceedings of IEEE Symposium on Foundations of Computer Science.* IEEE Computer Society, 1982.

[Kis82] G. Kissin. Measuring Energy Consumption in VLSI Circuits: a Foundation. In *Proceedings of ACM Symposium on Theory of Computing.* ACM-SIGACT, 1982.

[Kis85] G. Kissin. Functional Bounds on Switching Theory. In *Chapel Hill Conference on VLSI.* U.N.C., Chapel Hill, Computer Science Press, 1985.

[Kis87] G. Kissin. *Measuring Energy Consumption in VLSI Circuits.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, 1987. Available as Toronto Computer Science Technical Report 202/87.

[KZ81] Z. Kedem and A. Zorat. On Relation between Input and Communication / Computation in VLSI. In *Proceedings of IEEE Symposium on Foundations of Computer Science.* IEEE Computer Society, 1981.

[Leo84] J. Leo. Energy Complexity in VLSI. Master's thesis, Informatics Department, Nijmegen University, The Netherlands, February 1984.

[LM81] T. Lengauer and K. Mehlhorn. On the Complexity of VLSI Computations. In *Proceedings of CMU Conference on VLSI - VLSI Systems and Computations*, pages 89–99. CMU, Computer Science Press, 1981.

[LS81] R.J. Lipton and R. Sedgewick. Lower Bounds for VLSI. In *Proceedings of ACM Symposium on Theory of Computing.* ACM-SIGACT, 1981.

[MC80] C. Mead and L. Conway. *Introduction to VLSI Systems.* Addison-Wesley, Reading, Mass., 1980.

[PRS81] M. S. Paterson, W. L. Ruzzo, and L. Snyder. Bounds on Minimax Edge Length for Complete Binary Trees. In *Proceedings of ACM Symposium on Theory of Computing.* ACM-SIGACT, 1981.

[PS82] C. Papadimitriou and M. Sipser. Communication Complexity. In *Proceedings of ACM Symposium on Theory of Computing.* ACM-SIGACT, 1982.

[ST86] L. Snyder and A. Tyagi. The Energy Complexity of Transitive Functions. Technical Report TRCS-86-09-07, Dept. of Computer Science, University of Washington, Seattle, 1986. A preliminary version appears in the Proceedings of the Twenty-fourth Annual Allerton Conference on Communication, Control and Computing, 1986.

[Tho79] C.D. Thompson. Area-Time Complexity for VLSI. In *Proceedings of ACM Symposium on Theory of Computing.* ACM-SIGACT, 1979.

[Tho80] C. D. Thompson. *A Complexity Theory for VLSI.* PhD thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1980.

[TR90] A. Tyagi and J. Reif. Energy Complexity of Optical Computations. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing.* IEEE, December 1990. to appear.

[Tya88] A. Tyagi. *The Role of Energy in VLSI Computations.* PhD thesis, Department of Computer Science, University of Washington, Seattle, 1988. *Available as UWCS Technical Report Number 88-06-05.*

[Tya89] A. Tyagi. Energy-Time Trade-Offs in VLSI Computations. In *Proceedings of the Ninth Conference on Foundations of Software Technology & Theoretical Computer Science*, pages 301–311. Lecture Notes in Computer Science #405, Springer-Verlag, 1989. *An extended version submitted to JCSS.*

[Ull84] J.D. Ullman. *Computational Aspects of VLSI.* Computer Science Press, Rockville, Md., 1984.

[Vui83] J. Vuillemin. A Combinatorial Limit to the Computing Power of VLSI Circuits. *IEEE Transactions on Computers*, pages 294–300, March 1983.

[WE85] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design.* Addison-Wesley, Reading, Mass., 1985.

[Yao79] A.C. Yao. Some Complexity Questions Related to Distributed Computing. In *Proceedings of ACM Symposium on Theory of Computing.* ACM-SIGACT, 1979.

[Yao81] A.C. Yao. The Entropic Limitations on VLSI Computations. In *Proceedings of ACM Symposium on Theory of Computing.* ACM-SIGACT, 1981.

Approximating the Mean Waiting Time Under a TDMA Schedule

Thomas .K. Philips
IBM Research Division, T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598

and

Shivendra S. Panwar
Department of Electrical Engineering and Computer Science,
Polytechnic University, 333 Jay Street, Brooklyn NY 11201

ABSTRACT

We examine the mean waiting time of a packet that is scheduled according to a periodic Time Division Multiple Access (TDMA) schedule. An exact solution method whose computational and memory requirements grow cubically and quadratically respectively in the problem size is known [2]. We present a simple but accurate approximation whose computational and memory requirements are linear in the problem size. For the relatively small example presented, the computational requirements are reduced by a factor of 10^5, and the memory requirements are reduced by a factor of 70.

I. INTRODUCTION

Hofri and Rosberg [2] have analyzed queueing delays in periodic TDMA control policies for the Multiple Access channel in which users are allowed access as follows. The time axis is assumed to be *slotted*, and users transmit messages that can be broken up into an integral number of constant length packets, each of which can be transmitted in a slot. A *transmission schedule* is defined for the channel and is known to each user. The transmission schedule is a mapping from the set of all users onto the set of all available slots. The transmission schedule is *periodic,* i.e., if slot i is allotted to a given user, then slot $i+N$ is allotted to the same user. N is called the *period* of the schedule, and is the same for every link. Slots are preassigned to users, and a slot is used if and only if a packet from its preassigned user awaits transmission. Unassigned slots are not ever used. All the users are synchronized, i.e. they agree on the start of every slot, and no collisions are ever caused by two or more users attempting to use the same slot. As the schedule is periodic, we restrict our attention to a single *transmission cycle,* which is defined to be a block of N contiguous slots. Furthermore, the i^{th} user is assigned X_i slots in each transmission cycle. As users do not ever share slots, their queues are independent, and we can focus our attention on a single user, say user 1, who is assigned X_1 slots. Starting at some arbitrary slot assigned to user 1, and following a transmission cycle in the

---

This work was supported in part by the New York State Center for Advanced Technology in Telecommunications, Polytechnic University, Brooklyn, NY 11201, and by the National Science Foundation under Grant NCR-8909719.

clockwise direction, the interpermit distances between slots assigned to user 1 are denoted d_j, $1 \leq j \leq X_1$. Clearly, $1 \leq d_j \leq N$ and $\sum_{j=1}^{X_1} d_j = N$. The arrival process is assumed to be Poisson, with arrival rate λ packets *per slot*. The utilization, ρ, is given by $\frac{\lambda N}{X_1}$.

Under the above policy, expressions for the mean queue length at the beginning of a random slot are derived in [2]. The expression, however, is complex. Its computational requirements are $O(X_1^3)$ and its memory requirements are $O(X_1^2)$. Consequently, this exact solution is not very useful for quick "back of the envelope" calculations. To put this in perspective, generating the results for the figures in this paper (with X_1 being limited to 89) took over 12 hours of CPU time on an IBM 3090 mainframe, and required the use of over 150 Kilobytes of memory. Our approximation, in contrast, requires $O(X_1)$ time and space, and for the same problem, it took about one second of CPU time and used about 2 Kilobytes of memory. Furthermore, in the limit as $\rho \to 0$, our approximation becomes exact. The advantage of the approximation over the exact method grows as X_1 increases.

The paper is organized as follows. In section 2, some exact computational results using the algorithm in [2] are presented. These results are then used to motivate our approximation. The approximation is then compared to the exact solution. Finally, in Section 3, conclusions are drawn and some open problems identified.

II. ANALYSIS

Hofri and Rosberg [2] show that the mean queue lenght at the beginning of a random slot (not necessarily a slot assigned to the user of interest) under the periodic TDMA policy described in the introduction is given by

$$\overline{L} = \frac{(\sigma^2 + \lambda)N + \sum_{j=1}^{X_1} \lambda d_j(\lambda d_j - 2(1 - p_j)) - 2\sum_{j=1}^{X_1}(1 - \lambda d_j)\sum_{k=1}^{j-1}(\lambda d_k - (1 - p_k))}{2(X_1 - \lambda N)} \qquad (1)$$

$$+ \frac{1}{N}\sum_{j=1}^{X_1} d_j \sum_{k=0}^{j-1}(\lambda d_k - (1 - p_k)) + \frac{1}{N}\sum_{j=1}^{X_1}(d_j - 1)\left[\frac{\lambda d_j}{2} - (1 - p_j)\right],$$

where λ is the mean number of arrivals *per slot*, σ^2 is the variance of the number of arrivals in a slot, $p_i = (X_1 - \lambda N)\eta_i$ and η_i is the i^{th} component of η, the left eigenvector corresponding to the largest eigenvalue of the stochastic matrix Γ, where Γ is defined via

$$\Gamma = \sum_{k \geq 0} H_k \Gamma^k. \qquad (2)$$

$H(k)$, in turn, is defined to be

$$\begin{bmatrix} 0 & q_1(k) & 0 & \cdots & 0 \\ 0 & 0 & q_2(k) & \cdots & \vdots \\ \vdots & \vdots & 0 & \cdots & 0 \\ 0 & \vdots & \vdots & \cdots & q_{X_1-1}(k) \\ q_{X_1}(k) & 0 & \vdots & & 0 \end{bmatrix}, \qquad (3)$$

where $q_i(k)$ is the probability that there are k arrivals in d_i slots.

If the arrivals are Poisson, the queue length seen by a random arrival is given by $\overline{L} + \frac{\lambda}{2}$, and from Little's law, it follows that $\overline{T}$, the mean time spent in the system by a random arrival is

$$\overline{T} = .5 + \frac{\overline{L}}{\lambda}. \tag{4}$$

There is a typographical error in [2] that is corrected here. In the last term of equation (1) the terms d_j and $d_j - 1$ have been transposed. This is best seen by following the derivation in [2]. In the special case when $X_1 = N$, the last term should vanish.

Looking at equations (1) to (3) it is clear that substantial amounts of computation are required to solve these equations. As raising Γ to various powers involves matrix multiplication, the computational requirements are $O(X_1^3)$. Γ is a dense matrix, and consequently requires $O(X_1^2)$ storage. Moreover, the implied constants are large. Most of the time is spent solving equation (2). As no direct solution for equation (2) exists, it has to be solved iteratively. As an indication of the computational requirements of this set of equations, we note that it took over twelve hours of CPU time on an IBM 3090 mainframe to generate the figures for this paper. Furthermore, the expression does not simplify under assumptions of light or heavy traffic. Our approximation seeks to address these limitations of the exact solution.

In order to motivate our approximation, we note that in most practical applications it is desirable that the slots allotted to a user be fairly evenly spaced. As a perfectly even spacing is not always possible (for example set $N = 5$ and $X_1 = 2$), we focus our attention on a slot assignment that is

1. Reasonably uniform, and
2. Easily reproducible.

Such an assignment can be generated using the Golden Ratio policy of Itai and Rosberg [1]. Let $\phi^{-1} \triangleq \frac{\sqrt{5} - 1}{2} = .6180339\ldots$ ϕ is known as the Golden Ratio, and is related to the Fibonacci numbers via

$$F_k = \frac{\phi^k - (1 - \phi)^k}{\sqrt{5}}. \tag{4}$$

We assume that there are S users, and that user i is assigned X_i slots. Without loss of generality, we can set N, the length of the transmission cycle to $\sum_i X_i$, as any unused slots can simply be assigned to a dummy user. Following [1], mark off the points $\phi^{-1} \bmod 1$, $2\phi^{-1} \bmod 1$, ..., $N\phi^{-1} \bmod 1$ on a circle of circumference 1, dividing it into N intervals. Allot to user 1 the slots corresponding to $\phi^{-1} \bmod 1$, $2\phi^{-1} \bmod 1$, ..., $X_1\phi^{-1} \bmod 1$, to user 2 to those corresponding to $(X_1 + 1)\phi^{-1} \bmod 1$, ..., $(X_1 + X_2)\phi^{-1} \bmod 1$, and so on till all the slots have been allotted.

Consider the following example: $S = 3$, $N = 8$, $X_1 = X_2 = 3$, $X_3 = 2$. The transmission schedule is 2 1 3 2 1 2 1 3 ..., as can be verified by direct computation. Note that successive transmission permits to a user are fairly evenly spaced. Define F_k to be the largest Fibonacci number that is less than or equal to N, and F_{k_1} to be the largest Fibonacci number that is less than or equal to X_1. In [1] it is shown that the distances between successive permits to a user can take on only three different values: F_{k-k_1}, F_{k-k_1+1} and F_{k-k_1+2}. Furthermore, there is at least one interval of length F_{k-k_1+2} and one of length F_{k-k_1+1}. The longest interpermit interval is therefore roughly 2.6 times as long as the shortest one. This assures that the sequence is not too uniform (in which case even a very simple minded approximation would do very well).

To examine the effects of varying the number of slots allotted to a user, we fixed N to be 89 (a Fibonacci number) and allowed X_1 to vary between 1 and 89. The results of the computation are

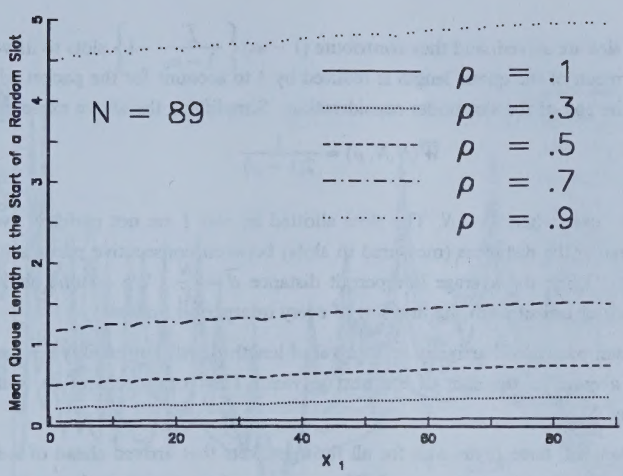


Figure 1. The Mean Queue Length at the Start of a Random Slot (Golden Ratio Assignment)

shown in Figure 1, which shows the mean queue length at the start of a random slot as a function of X_1 and the utilization ρ. Note that the queue length depends strongly on ρ and only weakly on X_1. This is the key to our approximation. We solve for the queue length exactly when $X_1 = N$, and then use this result to approximate the queue length for all other values of X_1. A word of caution is in order at this point- the fact that we assume the queue lengths to be same for all values of X_1 does *not* imply (via Little's law) that our approximation to the mean response time is independent of X_1. Other effects make the mean waiting time depend strongly on X_1.

When $X_1 = N$, the system reduces to a discrete time M/D/1 queue, and the following results can be found in [3].

1. The queue length at the start of a random slot has a steady state distribution $\pi = \pi_0, \ \pi_1, \ldots$
2. $\pi_0 = 1 - \lambda = 1 - \rho$
3. The mean queue length at start of a random slot, $\overline{L}$,is given by $\dfrac{\lambda[2-\lambda]}{2[1-\lambda]} = \dfrac{\rho[2-\rho]}{2[1-\rho]}$.

Define $\overline{W}(X, N, \rho)$ to be the mean waiting time (in slots) for a random arrival when X slots are allotted to a user. We can write

$$\begin{aligned} \overline{W}(N,N,\rho) &= .5 + \frac{\lambda}{2} + (1-\pi_0)\left\{\frac{\overline{L}}{1-\pi_0} - 1\right\} \\ &= .5 + \frac{\rho}{2} + (1-\pi_0)\left\{\frac{\overline{L}}{1-\pi_0} - 1\right\}. \end{aligned} \tag{5}$$

The terms arise as follows. On average, a packet arrives in the middle of a slot and this contributes half a slot to its waiting time. Again, on average, $\frac{\lambda}{2}$ packets arrive ahead of it in the same slot, and it has to wait for them to be served. In addition, it has to wait till all the packets enqueued at the beginning of the slot are served, and they contribute $(1-\pi_0)\left\{\frac{\bar{L}}{1-\pi_0}-1\right\}$ slots to its waiting time. The conditional mean of the queue length is reduced by 1 to account for the packet whose service is completed at the end of the slot under consideration. Simplifying the above expression yields

$$\overline{W}(N,N,\rho)=\frac{1}{2(1-\rho)}\,. \tag{6}$$

Next consider the case when $X_1 < N$. The slots allotted to user 1 are not perfectly evenly spaced now. As stated earlier,the distances (measured in slots) between consecutive permits to user 1 are denoted $d_1, \dots d_{X_1}$. Define the average interpermit distance $\bar{d}=\frac{N}{X_1}$. We assume, with no loss of generality, that a user transmits in the first slot of every interpermit interval.

A randomly chosen packet will arrive in an interval of length d_i with probability $\frac{d_i}{N}$, and will wait $\frac{d_i}{2}$ slots on the average to the start of the next service. This component of its waiting time is therefore given by $\sum_{i=1}^{X_1}\frac{d_i^2}{2N}$.

The chosen packet will have to to wait for all those packets that arrived ahead of it in the same interval to be served before it can be served. The mean number of packets that arrive ahead of it in an interpermit interval of length d_i is given by $\lambda d_i/2$. We shall assume that the distance between slots assigned to user 1 is exactly $\bar{d}$, and consequently there is an additional contribution to the mean waiting time of

$$\sum_{i=1}^{X_1}\frac{d_i}{N}\times\frac{\lambda d_i}{2}\times\bar{d} = \sum_{i=1}^{X_1}\frac{d_i^2\rho}{2N}\,. \tag{7}$$

Finally, we approximate the delay due to customers enqueued at the start of the interval in which the chosen packet arrived. The mean queue length at the start of every interval is assumed to be given by $\rho(2-\rho)/2(1-\rho)$, and the probability of the queue being empty at the start of any interval is assumed to be $1-\rho$. Once again, we assume that the interpermit intervals are spaced by exactly $\bar{d}$ slots, and following the approach developed in equation (5), we see that this contributes $\bar{d}\rho^2/2(1-\rho)$ slots to the mean waiting time. On collecting terms and simplifying, we get

$$\overline{W}(X_1,N,\rho)=(1+\rho)\sum_{i=1}^{X_1}\frac{d_i^2}{2N}+\frac{\bar{d}\rho^2}{2(1-\rho)} \tag{8}$$

In the special case when all the interpermit intervals are equally long, (8) is exact. Furthermore, for an arbitrary collection of interpermit intervals, it becomes exact as $\rho \to 0$, for in that case all the terms involving the utilization and the mean interpermit distance disappear. The expression was compared to the computed solution and the error in the approximation is plotted in Figure 2 for utilizations ranging from .1 to .9. From the figure we see that our approximation tends to over estimate the mean waiting time, and that the maximum error does not exceed 2.5%. The variation in interpermit distances is minimized when X_1 is a Fibonacci number (see Theorem 5.1 of [2]) and in these cases the approximation is seen to more accurate. The expression is exact when $\lambda = 0$.

To further test the validity of the approximation, we generated a sequence of random slot assignments and compared the approximation with the exact solution for these slot assignments. Specifically, we maintained N at 89, and for each value of X_1 from 1 to 89, and for each value of ρ between

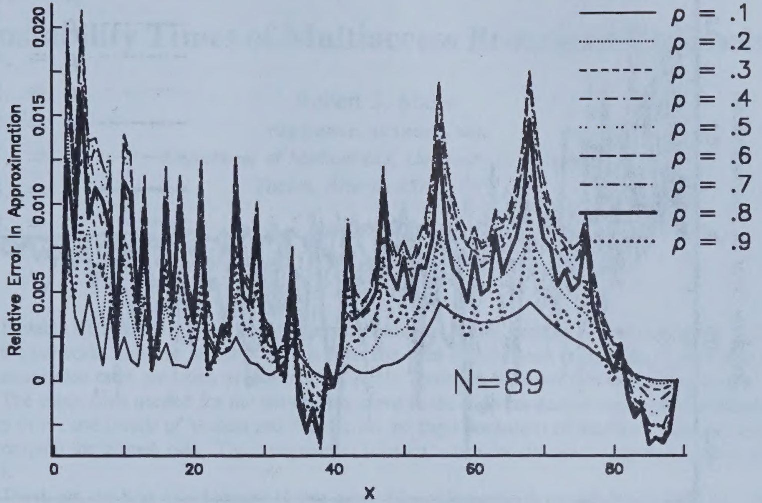


Figure 2. Error in mean waiting time (Golden Ratio Assignment)

.1 to .9 (with a spacing of .1), we selected a random subset of X_1 slots by sampling uniformly without replacement from the set of 89 slots, and computed both the exact solution and our approximation. Note that for a fixed value of X_1, a (possibly) different slot assignment is generated for each value of ρ. This has the advantage of testing the approximation over the widest possible range of slot assignments. The errors are still quite small, though they sometimes rise to about 20 percent or so, particularly when X_1 is small and the slots are very unevenly distributed. In any event, the error is still more than acceptable for most engineering purposes.

III. CONCLUSIONS

A simple approximation has been presented for the mean waiting time of a packet under a TDMA scheduling policy whose computational and memory requirements grow linearly with the problem size. The approximation becomes exact in light traffic. It would be of interest to see if the quality of the approximation could be further improved in heavy traffic by using limited information about the sequence of interpermit distances. This would allow the use of an interpolation approximation that interpolated between the (exact) light traffic result and the heavy traffic limit. Work on this is currently under way.

REFERENCES

1 . Itai, A., and Rosberg, Z., 'A Golden Ratio Control Policy for a Multiple Access Channel', IEEE Trans. Auto. Control, Vol. AC-29, 1984, pp 712-718.

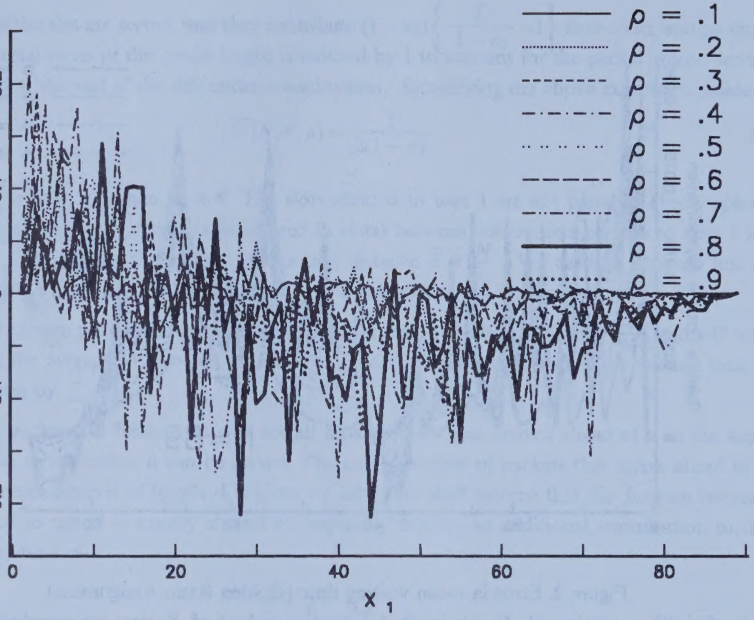


Figure 3. Error in mean waiting time (Random Slot Assignment)

2 . Hofri, M., and Rosberg, Z., 'Packet Delay under the Golden Ratio Weighted TDM Policy in a Multiple-Access Channel', IEEE Trans. on Info. Theory, Vol. IT-33, 1987, pp 341-349.

3 . Kobayashi, H., 'Discrete Time Queueing Systems', IBM Research Report RC 8817, 1981, pp 8-9.

Instability Times of Multiaccess Broadcast Channels*

Robert S. Maier
`rsm@math.arizona.edu`
Department of Mathematics, University of Arizona,
Tucson, Arizona 85721, USA

Abstract

Bistable CSMA/CD and ALOHA systems are studied, in the limit as N, the number of nodes on the network, tends to infinity. In this limit the total offered load is held fixed, and packet retransmission rates are taken proportional to N^{-1}. There are typically two congestion points.

The mean time needed for the network to move to the high-congestion state grows exponentially in N, and results of Ventcel and Freidlin on the large deviations of Markov chains are used to compute the growth rate. The computation is exact, since the diffusion approximation is not used.

These are the first exact results in this area. The technique can readily be applied to other systems, in particular networks employing protocols with retransmission counters.

Keywords: random access algorithms, communications protocols, multiaccess protocols, Markov chains, first exit times, large deviations

1. Introduction

Many authors have explored the appearance of instabilities in random access broadcast protocols, such as ALOHA or CSMA/CD. A heavily loaded network, when modelled probabilistically, frequently develops *bistable behavior*: under conditions of heavy load, the associated Markov chain exhibits two points of stable equilibrium. One corresponds to an desirable high-throughput, low-congestion state, and the other to an undesirable low-throughput, high-congestion state. A transition from the former congestion point to the latter will cause a sudden degradation in network performance.

Let N be the number of nodes on the network. Then in order to examine the transient behavior of a well-populated network, one must take an appropriate $N \to \infty$ limit. This requires care. Of course as $N \to \infty$ the network load, the total number of offered packets per unit time, should be held fixed. The arrival rate at each node will accordingly be proportional to N^{-1}. Many authors assume that packet transmission strategies (*e.g.*, per-node retransmission rates) are independent of N as well. This assumption yields a well-defined $N = \infty$ ("infinite user") model, and whether or not this model is stable has been extensively investigated [**4**, **9**]. But it is not the only possible limiting model.

Even though holding transmission strategies fixed as $N \to \infty$ has a certain plausibility — after all, they are typically wired into the hardware — it yields a comparatively uninteresting limit. In order to preserve the bistable behavior for arbitrarily large N, and obtain an $N = \infty$ limit characterised by two stable fixed points, one must take the retransmission rates proportional to N^{-1} [**5**].

In this limit one can study the $N \to \infty$ asymptotics of the amount of time needed for the transition to the high-contention state to take place, the so-called First Exit Time (FET) [**11**]. The mean FET is

---

*This work was supported in part by AFOSR contract 88-0189.

the strongest measure of the degree of instability of a finite-N network. Most studies of stability are qualitative, but the mean FET provides a *quantitative* measure of network stability.

In the bistable region the mean FET will, to leading order, grow exponentially in N. Numerical estimation of the growth rate is difficult, since the network state space grows rapidly in size as $N \to \infty$. (In fact if the nodes are assumed to buffer n packets, and to exist in one of m possible 'retransmission states,' then the state space will have $(mn+1)^N$ states.) A theoretical treatment is necessary. The strongest results to date are those of Cottrell *et al.* [**1**] and Günther and Shaw [**5**], both of whom analysed slotted ALOHA models with $m = 1$ and $n = 1$.

The present author has employed modern probabilistic tools to compute the exact exponential growth rate. There is a well-developed theory of the large fluctuations of Markov chains, and the asymptotics of their exit times [**2, 12, 13**]. It goes under the name of Ventcel-Freidlin theory, and has been applied to the long-time evolution of data structures [**8**] and a simple model of storage allocation [**7**]. Weiss [**14**] has applied Ventcel-Freidlin theory to problems of flow control and buffer overflow, and Lagarias and Weiss [**6**] have used it to study diffusion and branching processes suggested by the $3x+1$ Problem. If the first exit time is denoted τ, then according to the theory its expectation has leading-order asymptotics

$$\mathsf{E}\{\tau\} \sim C N^{\alpha} e^{N S_0}, \qquad N \to \infty$$

in which S_0 can be computed exactly. Computing C and α is more difficult.

The Ventcel-Freidlin formalism is applied below to two simple random access systems: a slotted ALOHA model and an unslotted persistent CSMA/CD model, both with $m = 1$ and $n = 1$. In both cases the worst-case star topology is assumed. The extension to more sophisticated protocols is now in progress.

2. The Models

2.1. The Slotted ALOHA Model

The time-slotted ALOHA model to be considered is standard. It comprises a single channel and N nodes, to each of which is attached an independent input process generating packets. It is assumed that on account of flow control, each process generates at most one new packet by each *slot time* $j = 0, 1, 2, \ldots$ Since N is large, the network-wide packet arrival process is assumed to be Poisson.

Between successive slot times, there is probability p_0 of a new packet appearing at each node. At each slot time $j > 0$, each node which has just received a packet will transmit it. Each node receives channel feedback as to whether or not a collision occurs, and deletes the packet from its queue only if it was successfully transmitted.

If a collision occurs, retransmission will be attempted later. In fact in each successive time slot there will be probability p_1 of its being retransmitted. Repeated attempts will be made, without bound, until success is attained. While the packet is being stored for retransmission, no new packets will be accepted.

This model is 'non-topological' in the sense that all nodes are equivalent; not only are packets transmitted synchronously, but simultaneously transmitted packets are assumed always to interfere with each other. The state of the system at slot time j is for all practical purposes specified by $X(j)$, the number of nodes that are currently blocked. $X(j)$ is a Markov chain on $[0, N]$; it is naturally expressed in a tail-recursive way as

$$X(j) = \begin{cases} X_0 & \text{if } j = 0, \\ X_0 + \sum_{k=0}^{j-1} \Delta_k(X(k)) & \text{if } j \geq 1. \end{cases} \tag{1}$$

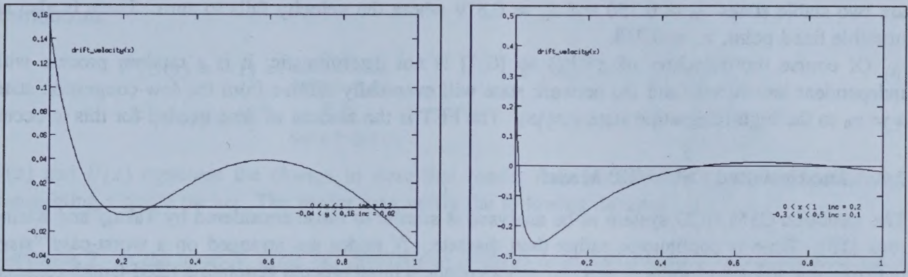


Figure 1: (a) The expected drift velocity $\mathsf{E}\{\Delta(x)\}$ of the $m = 1$, $n = 1$ ALOHA model, as a function of normalised network state x. Parameter values are $q_0 = 0.43$ and $q_1 = 5.0$. (b) The expected drift velocity of the $n = 1$ CSMA/CD model. Parameter values are $q_0 = 0.4$, $q_1 = 40$, $a = 0.1$ and $b = 0.5$.

Here X_0 denotes the initial network state, and $\Delta(\cdot)$ is the state change from time-slot to time-slot. It is a random variable whose distribution is parametrised by the current network state; the subscript k merely signifies that the summands in (1) are independent instances.

The distribution of $\Delta(\cdot)$ is easy to determine in the large-N limit. As noted in Section 1, one should take $p_0 = N^{-1}q_0$, with q_0 the total network load. Moreover one should choose $p_1 = N^{-1}q_1$, for some constant $q_1 > 0$, to make the retransmission probabilities scale also. As will become clear in Section 3, for large N one should study the behaviour of the *normalised* network state

$$x^{(N)}(t) \triangleq N^{-1}X(\lfloor Nt \rfloor). \tag{2}$$

Here t is a 'normalised time,' differing by a factor N from the slot time j. For any N, $x^{(N)}(t)$ is a continuous-time jump process, and as t increases it moves through the *normalised* network state space $[0, 1]$. The definition (2) "rescales space and time equally," so that drift velocities are preserved.

This choice of scaling ensures that the mean number of nodes performing each type of action will converge as $N \to \infty$. Let A be the number of nodes at normalised time t that transmit a newly arrived packet, and R be the number of nodes that retransmit a packet. Then in the limit A and R become independent Poisson random variables, with means $q_0(1 - x(t))$ and $q_1 x(t)$ respectively. The following theorem on $\Delta(x)$, the state change as a function of normalised state, follows by inspection.

THEOREM 1. *The random variable* $\Delta(x)$ *equals* $A(x) - (A(x) + R(x))\delta_{A(x)+R(x),1}$.

COROLLARY.

$$\mathsf{E}\{\Delta(x)\} = q_0(1 - x) - [q_0(1 - x) + q_1 x]e^{-[q_0(1-x)+q_1 x]}. \tag{3}$$

Equation (3) appears, in slightly different notation, in Günther and Shaw [5]. They note that if q_0 and q_1 are appropriately chosen, the equation

$$\mathsf{E}\{\Delta(x)\} = 0$$

will have more than a single solution x. Figure 1(a) graphs $\mathsf{E}\{\Delta(x)\}$ as a function of x, for $q_0 = 0.43$ and $q_1 = 5.0$ (the values being taken from [5]). The presence of two solution signals *bistability*: in the fluid approximation $\mathsf{E}\{\Delta(x)\}$ is interpreted as a state-dependent *drift velocity* on $\mathcal{Q}$, and there are two stable states $x_0 \approx 0.150$ and $x_1 \approx 0.879$ where the velocity falls to zero. There is also an unstable fixed point, $x_c \approx 0.278$.

Of course the trajectory of $x^{(N)}(t)$ on $[0, 1]$ is not deterministic; it is a random process with independent increments, and the network state will eventually diffuse from the low-congestion state $x = x_0$ to the high-congestion state $x = x_1$. The FET is the amount of time needed for this to occur.

2.2. The Unslotted CSMA/CD Model

The unslotted CSMA/CD system to be analysed is similar to those considered by Takagi and Kleinrock [**10**]. Time is continuous, rather than discrete. N nodes are arranged on a worst-case "star" network, and to each of them is attached an independent input process generating fixed-length packets. On the average, p_0 packets arrive per unit time. By convention the packets are 1 time unit long. Upon arrival of a packet, it is immediately transmitted. Unless, that is, the node recognises the transmission channel as busy, in which case the new packet is simply enqueued. It is assumed that flow control allows at most one new packet to arrive at each node while the channel is busy.

Because the network is star-shaped, propagation delay between any pair of nodes is some constant a time units; by assumption $a < 1$. On account of this delay, one or more nodes may begin to transmit within a time a after a transmission has begun. If so a collision occurs, and each node detecting the collision transmits a jamming tone. Again on account of propagation delays, it requires $b \geq a$ time units before the original transmission is suppressed, and a further a time units before the channel is universally recognized as clear. So unsuccessful busy periods occupy $Y+b+a$ time units ($Y < a$ being the time between the first transmission and the colliding one), and successful ones $1 + a$ time units.

If a packet is unsuccessfully transmitted, or arrives at a time when the channel is recognised as in use, it is not discarded but enqueued. Retransmission will be attempted later. After the channel reopens, nodes back off exponentially: the time to retransmission is an exponential random variable, of mean $1/p_1$.

As in the slotted ALOHA model, one should scale p_0 and p_1 as $N \to \infty$ by choosing $p_0 = N^{-1}q_0$ and $p_1 = N^{-1}q_1$, for q_0 and q_1 constant. With this scaling, one should study the embedded Markov chain obtained by taking a snapshot of the system after each busy period. The state of the system after the jth busy period may be denoted $X(j)$; it is the number of blocked nodes. The distribution of Δ, *i.e.*, $X(j+1) - X(j)$, depends on the current state $X(j)$, or equivalently the current *normalised* state $x(t) = N^{-1}X(\lfloor Nt \rfloor)$; it may be worked out as follows.

After the channel reopens, the Nx blocked nodes transmit at a rate $N^{-1}q_1$, and the $N(1 - x)$ blocked nodes transmit newly arrived packets at a rate $N^{-1}q_0$. Suppose one or the other sort of node begins to transmit a packet. There will be a 'dangerous interval' of length a during which an additional transmission may occur and cause a collision. Since for large N the network-wide transmission process is Poisson, a collision occurs with probability $e^{-a[q_0(1-x)+q_1 x]}$.

Suppose that the initial packet is transmitted successfully. Since a successful busy period is of length $1 + a$, there is 1 additional time unit in which packets may arrive and be enqueued. On the other hand, if a collision occurs then the transmission period will last for b additional time units. So one defines A_1 to be the number of packets that arrive and are enqueued during the additional 1 time unit of a *successful* busy period, and A_b to be the number of packets that arrive and are enqueued during the additional b time units of an *unsuccessful* busy period. In the large-N limit, A_1 and A_b become independent Poisson random variables, with means $q_0(1 - x)$ and $q_1(1 - x)b$ respectively.

Of course one must take into account the fact that the initial packet, and the colliding one if any, may be either a newly arrived packet or a previously queued packet. To do this one defines random variables $S(x)$ and $U(x)$, which are independent of each other and of A_1 and A_b, and which have distributions

$$\mathsf{P}\{S(x)=-1\}=\frac{q_1x_1}{q_0x_0+q_1x_1} \qquad \mathsf{P}\{S(x)=0\}=\frac{q_0x_0}{q_0x_0+q_1x_1} \tag{4}$$

$$\mathsf{P}\{U(x)=0\}=\frac{q_1x_1}{q_0x_0+q_1x_1} \qquad \mathsf{P}\{U(x)=1\}=\frac{q_0x_0}{q_0x_0+q_1x_1}. \tag{5}$$

$S(x)$ and $U(x)$ represent the change in state that results from successfully (resp. unsuccessfully) transmitting a single packet. The reader may verify the following theorem.

THEOREM 2. *In the large-N limit, the distribution of $\Delta(x)$ is identical to that of a random variable which equals*

$$\begin{cases} S(x)+A_1(x), & \textit{with probability } e^{-a[q_0(1-x)+q_1x]} \\ U_1(x)+U_2(x)+A_b(x), & \textit{with probability } 1-e^{-a[q_0(1-x)+q_1x]}. \end{cases}$$

Here $U_1(x)$ and $U_2(x)$ are independent instances of $U(x)$.

COROLLARY.

$$\begin{aligned} \mathsf{E}\{\Delta(x)\} \;=\; & q_0b(1-x)+\tfrac{2q_0(1-x)}{q_0(1-x)+q_1x} \\ & +\left(q_0(1-b)(1-x)-1-\tfrac{q_0(1-x)}{q_0(1-x)+q_1x}\right)e^{-a[q_0(1-x)+q_1x]} \end{aligned} \tag{6}$$

Figure 1(b) graphs $\mathsf{E}\{\Delta(x)\}$ as a function of x, for typical choices of the parameters: $q_0=0.4$, $q_1=40$, $a=0.1$ and $b=0.5$. As with the $n=1$ ALOHA $\mathsf{E}\{\Delta(x)\}$ graphed in Figure 1(a), there are two stable fixed points $x_0\approx 0.010$ and $x_1\approx 0.862$, and an unstable fixed point $x_c\approx 0.505$, where the drift velocity falls to zero.

But Figure 1(b), like Figure 1(a), says nothing about the amount of time needed to diffuse from x_0 to x_1, or about the asymptotics of its mean as $N\to\infty$. Computing the growth rate of the mean FET can be accomplished, however, by the techniques of the next section.

3. Ventcel-Freidlin Theory

The natural setting for the discussion of mean FETs is a family of Markov chains $\{\mathrm{x}^{(N)}(t)\}_{N=1,2,\dots}$ on a normalised state space $\mathcal{Q}\subset\mathbb{R}^d$. The increments of the Markov chains should be independent and equal to $N^{-1}\Delta$, with Δ a random vector whose distribution is parametrised by the current normalised state x. In the ALOHA and CSMA/CD models of Sections 2.1 and 2.2, d equalled unity, $\mathcal{Q}$ was $[0,1]$, and the dependence of Δ on x was worked out explicitly.

In general $\mathcal{Q}$ will contain one or more stable fixed points. Fixed points are defined as normalised states $\mathrm{x}\in\mathcal{Q}$ satisfying $\mathsf{E}\{\Delta(\mathrm{x})\}=\mathbf{0}$. In the fluid approximation the normalised state simply flows toward the stable fixed points, which serve as attractors. This deterministic flow can be thought of as a *dynamical system*.

It is possible to compute the $N\to\infty$ asymptotics of the mean FET, *i.e.*, the time needed for the normalised state to 'swim against the flow' and move from one stable fixed point to another. The theory is due to Ventcel and Freidlin [**2, 12, 13**].

Suppose that Δ's distribution is parametrised, sufficiently smoothly, by $\mathrm{x}\in B$, some closed domain in $\mathbb{R}^d$. Suppose also that the vectorfield $\mathsf{E}\{\Delta(\mathrm{x})\}$ equals zero at only a single point $\mathrm{x}_0\in\mathring{B}$, and points inward. Then their theorem on the exit of the $\{\mathrm{x}^{(N)}(t)\}_{N=1,2,\dots}$ from B requires the following auxiliary functions[1].

---

[1] The terms *H-function* and *L-function* are from [**2**]. Readers of Ventcel's earlier papers [**12, 13**] should be aware that H and L were originally denoted G and H.

Definition. The *H-function* or *Hamiltonian* $H\colon\ B \times \mathbb{R}^d \longrightarrow \mathbb{R}$ associated to the family of random processes $\{\mathbf{x}^{(N)}(t)\}_{N=1,2,\ldots}$ is defined by

$$H(\mathbf{x},\mathbf{p}) \triangleq \log \mathsf{E}\left\{e^{\mathbf{p}\cdot\Delta(\mathbf{x})}\right\}. \tag{7}$$

$H(\mathbf{x},\cdot)$ is the generating function of the cumulants of $\Delta(\mathbf{x})$, and is a convex function.

Definition. The corresponding *L-function* or *Lagrangian* $L\colon\ B \times \mathbb{R}^d \longrightarrow \mathbb{R}$ is defined by

$$L(\mathbf{x},\mathbf{u}) \triangleq \sup_{\mathbf{p}\in\mathbb{R}^d} \left[\mathbf{p}\cdot\mathbf{u} - \log \mathsf{E}\left\{e^{\mathbf{p}\cdot\Delta(\mathbf{x})}\right\}\right]. \tag{8}$$

$L(\mathbf{x},\cdot)$ is the Legendre transform of $H(\mathbf{x},\cdot)$. For any $\mathbf{x}$, $L(\mathbf{x},\cdot)$ is a convex function.

The terms *Hamiltonian* and *Lagrangian* are taken from classical mechanics [3]. The following terminology is also taken from classical mechanics.

Definition. The *action* $\mathcal{S}(\cdot)$, a real-valued functional, is defined by

$$\mathcal{S}(\mathbf{x}(\cdot)) \triangleq \int_{s=0}^{T} L(\mathbf{x}(s),\dot{\mathbf{x}}(s))\,ds. \tag{9}$$

It is defined on all sufficiently smooth trajectories $\mathbf{x}\colon\ [0,T] \longrightarrow B$.

Definition. A *classical path* is a trajectory $\mathbf{x}\colon\ [0,T] \longrightarrow B$ that is a local extremum of the action $\mathcal{S}(\cdot)$. Equivalently, by the calculus of variations, it is a solution of the Euler-Lagrange equations

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{x}}} - \frac{\partial L}{\partial \mathbf{x}} = 0. \tag{10}$$

Definition. A *critical path* $\mathbf{x}^*: [0,T] \longrightarrow B$ is a classical path $\mathbf{x}\colon\ [0,T] \longrightarrow B$ that

1. satisfies $\mathbf{x}(0) = \mathbf{x}_0$ and $\mathbf{x}(T) \in \partial B$, and
2. minimises $\mathcal{S}(\mathbf{x}(\cdot))$ within the class of such functions, the minimum being taken over all $T > 0$ and $\mathbf{x}(T) \in \partial B$.

That is to say, it is a classical path from $\mathbf{x}_0$ to the boundary of B that minimises the action.

$$\mathcal{S}_0 \triangleq \mathcal{S}(\mathbf{x}^*(\cdot))$$

is the *critical action.* In control theory it would be termed an *optimal cost,* since $\mathcal{S}(\cdot)$ is a sort of cost function, associating a cost to each trajectory from $\mathbf{x}_0$ to ∂B.

Ventcel and Freidlin studied the extent to which the family of random processes $\{\mathbf{x}^{(N)}(t)\}_{N=1,2,\ldots}$ deviate from the deterministic flow, that is to say, from the trajectories $\mathbf{x}\colon\ [0,T] \longrightarrow B$ satisfying

$$\dot{\mathbf{x}} = \mathsf{E}\{\Delta(\mathbf{x})\}.$$

They showed that $\mathcal{S}(\cdot)$ serves as a *rate function* for the large deviations of this family: as $N \longrightarrow \infty$, the probability that $\mathbf{x}^{(N)}(t)$ follows any given trajectory $\mathbf{x}\colon\ [0,T] \longrightarrow B$ will be suppressed by a factor proportional to $e^{-N\mathcal{S}(\mathbf{x}(\cdot))}$. The precise statement is contained in their papers [**12, 13**]. They also proved the following closely related theorem on the *exit times* of the $\{\mathbf{x}^{(N)}(t)\}_{N=1,2,\ldots}$.

Definition. Let t_{exit} be the first time that $\mathbf{x}^{(N)}(t)$ leaves the interior of B, *i.e.*, $\inf\{t \geq 0 \mid \mathbf{x}^{(N)} \notin \mathring{B}\}$. The number of jumps that take place by time t_{exit}, *i.e.*, $\lfloor N t_{\text{exit}} \rfloor$, will be denoted τ.

THEOREM 3 (from [**2**]). *The mean exit time* $\mathsf{E}\{t_{\text{exit}}\}$ *satisfies*

$$\lim_{N\to\infty} N^{-1} \log \mathsf{E}\{t_{\text{exit}}\} = S_0$$

in which S_0 *is the critical action. That is to say, the critical action is the asymptotic exponential growth rate of* $\mathsf{E}\{t_{\text{exit}}\}$ *as* $N \to \infty$.

Remark. The same theorem holds with t_{exit} replaced by τ, since the factor N is subexponential.

They proved, moreover, that the critical path $\mathbf{x}^*(\cdot)$ is the most likely exit path. As $N \to \infty$, the probability that the exit path deviates significantly from it falls off exponentially in N. Typical escape trajectories behave as follows: they remain near $\mathbf{x}_0$ for an amount of time exponential in N, and then depart by the critical path. For large N, on the time scale of the escape from B the traversal of the critical path occurs all but instantaneously.

Theorem 3 is clearly a very powerful theorem. It relates the exponential growth rate of the mean exit time to the calculus of variations; in fact, to Hamilton-Jacobi theory. It allows one to compute the growth rate S_0 by standard variational arguments, which in the language of classical mechanics [**3**, **7**] run as follows.

In Hamiltonian mechanics a classical point particle has a *momentum trajectory*, $\mathbf{p}\colon [0,T] \to \mathbb{R}^d$, related to its velocity as a function of time by the two equivalent equations

$$\mathbf{p}(\mathbf{x},\dot{\mathbf{x}}) = \frac{\partial L}{\partial \dot{\mathbf{x}}}(\mathbf{x},\dot{\mathbf{x}}) \qquad \dot{\mathbf{x}}(\mathbf{x},\mathbf{p}) = \frac{\partial H}{\partial \mathbf{p}}(\mathbf{x},\mathbf{p}).$$

And the Hamiltonian $H(\mathbf{x},\mathbf{p})$ is regarded as the *energy* of the particle, as a function of position and momentum. It is guaranteed to be constant on any classical path $(\mathbf{x},\mathbf{p})\colon [0,T] \to \mathbb{R}^d \times \mathbb{R}^d$. For the critical path $(\mathbf{x}^*(t),\mathbf{p}^*(t))$ a bit more is known, since it is the solution of a variational problem that optimises not only the path between $\mathbf{x}_0$ and ∂B, but also the transit time.

THEOREM 4 ([**2**, **7**]). $H(\mathbf{x},\mathbf{p}) = 0$ *at all points on the critical path. That is to say, the critical path is a zero energy trajectory.*

By the definition of H, this theorem implies that

$$\mathsf{E}\left\{e^{\mathbf{p}\cdot\Delta(\mathbf{x})}\right\} = 1 \tag{11}$$

everywhere on the critical path. If $d = 1$ this equation can be solved implicitly for $p = p^*(x)$. The left-hand side is convex in p, and there will be exactly two solutions p for all $x \neq x_0$. One of them will be $p \equiv 0$, and the momentum of the critical path, as a function of position, will be the other. It turns out that knowledge of this function is sufficient to compute S_0. This is a standard result from classical mechanics.

THEOREM 5 (from [**3**]). *For all zero energy classical paths* $\mathbf{x}\colon [0,T] \to B$, *the action* $\mathcal{S}(\mathbf{x}(\cdot))$ *equals*

$$\int_{\mathbf{x}(\cdot)} \mathbf{p}(\mathbf{x}) \cdot d\mathbf{x}$$

in which the integral is a line integral along the classical path. In particular the critical action S_0 *satisfies*

$$S_0 = \int_{\mathbf{x}^*(\cdot)} \mathbf{p}^*(\mathbf{x}) \cdot d\mathbf{x}$$

the integral being taken along the critical path.

So if $d = 1$, S_0 can be evaluated with a minimum of effort. Suppose that $B = [x', x'']$, an interval containing x_0. The critical path will extend either from x_0 to x', or from x_0 to x''. One should solve (11) for $p = p(x)$, and evaluate the integrals

$$\int_{x_0}^{x'} p(x)\, dx \qquad \int_{x_0}^{x''} p(x)\, dx.$$

Whichever is less will be the critical action S_0, the asymptotic growth rate of t_{exit}.

4. FET Asymptotics

When properly normalised, stochastically modelled ALOHA and CSMA/CD systems are Markov chains of the sort considered in the last section: random processes $\{\mathbf{x}^{(N)}(t)\}_{N=1,2,\ldots}$ on a normalised state space $\mathcal{Q} \subset \mathbb{R}^d$. They begin at some initial point $\mathbf{x}_0$, and jump at deterministic times $t = N^{-1}, 2N^{-1}, 3N^{-1}, \ldots$ The jumps are of magnitude $N^{-1}\Delta(\mathbf{x})$, and the x-dependence of the random variable $\Delta(\mathbf{x})$ can be worked out, as in Theorems 1 and 2.

At one or more points in $\mathcal{Q}$, the expected drift velocity $\mathsf{E}\{\Delta(\mathbf{x})\}$ falls to zero. Normally there are at most two *stable* fixed points. The stable fixed point characterised by the higher network throughput will be denoted $\mathbf{x}_0$, and it will be used as the initial network state. The other will be denoted $\mathbf{x}_1$, and the FET is the time needed to reach it.

In general the simplex $\mathcal{Q}$ will be partitioned into *basins of attraction*: both $\mathbf{x}_0$ and $\mathbf{x}_1$ will draw towards themselves all points in some surrounding neighbourhood. There will typically be one or more hyperbolic fixed points as well, and the stable manifolds of these hyperbolic fixed points will separate the two basins of attraction.

So in the Ventcel-Freidlin framework, the set $B \subset \mathcal{Q}$, from which $\mathbf{x}(t)$ must escape, is the basin of attraction of $\mathbf{x}_0$. Theorem 3 guarantees that the asymptotic exponential growth rate of the mean FET will be S_0, the action of the critical path leading from $\mathbf{x}_0$ to the boundary ∂B.

By Theorems 4 and 5, in the $d = 1$ case S_0 can be computed from

$$M(\mathbf{x}, \mathbf{p}) \triangleq \mathsf{E}\left\{e^{\mathbf{p}\cdot\Delta(\mathbf{x})}\right\},$$

the generating function of $\Delta(\mathbf{x})$. The following propositions follow from the results of Section 2.

PROPOSITION 1. *In the slotted ALOHA system of Section 2.1, the generating function $M(x, p)$ of $\Delta(x)$ equals*

$$e^{q_0(1-x)(e^p-1)} + q_0(1-x)e^{-q_0-q_1x}(1-e^p) + q_1xe^{-q_0-q_1x}(e^{-p}-1).$$

PROPOSITION 2. *In the unslotted CSMA/CD system of Section 2.2, the function $M_{\mathit{CSMA/CD}}(x, p)$, the generating function of $\Delta(x)$, equals*

$$\begin{aligned} & e^{-a(q_0(1-x)+q_1x)} \left(\frac{q_0(1-x)+q_1xe^{-p}}{q_0(1-x)+q_1x}\right) e^{q_0(1-x)(e^p-1)} \\ + \quad & (1-e^{-a(q_0(1-x)+q_1x)}) \left(\frac{q_0(1-x)e^p+q_1x}{q_0(1-x)+q_1x}\right) e^{q_0b(1-x)(e^p-1)}. \end{aligned}$$

The expressions for $M(x, p)$ in Propositions 1 and 2 are easily handled numerically, and one can compute S_0 as sketched in the last section.

Example. Consider the $m = 1$, $n = 1$ slotted ALOHA system with the parameter values chosen by Günther [5] and used in Section 2: $q_0 = 0.43$ and $q_1 = 5.0$. The solution $p = p^*(x)$ of the implicit equation $M_{\mathrm{ALOHA}}(x, p) = 1$ may be worked out numerically. One obtains

$$\int_{x_0}^{x_c} p^*(x)\,dx \approx 0.00177 \qquad \int_{x_1}^{x_c} p^*(x)\,dx \approx 0.104$$

So by Theorems 3 and 5 one has the following results.

PROPOSITION 3. *For the $m = 1$, $n = 1$ slotted ALOHA system with parameters $q_0 = 0.43$ and $q_1 = 5.0$, the mean FET* $\mathsf{E}\{\tau\}$ *satisfies*

$$\lim_{N\to\infty} N^{-1} \log \mathsf{E}\{\tau\} = 0.00177$$

That is to say, to leading order the mean FET grows as $e^{0.00177N}$. Moreover, the time needed for the system to return from the high-congestion state grows as $e^{0.014N}$.

The final statement comes from Theorem 5, applied to the region $[x_c, 1]$ rather than to $[0, x_c]$. The difference in growth rates is striking: for large N it will be very difficult for the system to return from the high-congestion state x_1, once it has been entered.

Example. Consider the $n = 1$ unslotted CSMA/CD system with the parameter values used in Section 2: $a = 0.1$, $b = 0.5$, $q_0 = 0.4$ and $q_1 = 40$. As in the slotted ALOHA case, one may solve the implicit equation $M_{\mathrm{CSMA/CD}}(x, p) = 1$ for $p = p(x)$ numerically. One obtains

$$\int_{x_0}^{x_c} p^*(x)\,dx \approx 0.201 \qquad \int_{x_1}^{x_c} p^*(x)\,dx \approx 0.0467$$

So by Theorems 3 and 5 one has the following.

PROPOSITION 4. *For the $n = 1$ unslotted CSMA/CD system with parameters $a = 0.1$, $b = 0.5$, $q_0 = 0.4$ and $q_1 = 40$, the mean FET* $\mathsf{E}\{\tau\}$ *satisfies*

$$\lim_{N\to\infty} N^{-1} \log \mathsf{E}\{\tau\} = 0.201$$

That is to say, to leading order the mean FET grows as $e^{0.201N}$. Moreover, the time needed for the system to return from the high-congestion state grows as $e^{0.0467N}$.

So for large values of N there will be a great disparity between the mean FET and the time needed for the system to return to the low-congestion state. But in the CSMA/CD case, with this choice of parameter values, the difference is in the opposite direction from the ALOHA case: it is far easier for the system to return to normality than to leave it.

It should be pointed out that the interpretation of τ differs slightly between Propositions 3 and 4. In the former proposition it is a *slot time*: the time slot in which the normalised state first leaves the basin of attraction of the low-congestion state. In the latter it counts the steps of an embedded Markov chain. It is the number of *transmission periods* that take place before the boundary of the basin is reached.

5. Conclusions

Propositions 3 and 4 demonstrate the power of Ventcel-Freidlin theory. For the slotted ALOHA and unslotted CSMA/CD systems, it was possible to compute the exponential growth rate S_0 of the mean FET as N, the number of nodes on the network, tends to infinity.

The author is now extending this treatment in several directions; in particular, working out the dependence of S_0 on the system parameters. Multidimensional systems, with $m > 1$ or $n > 1$, are also being studied. The goal is the computation of the FET growth rate for realistic retransmission algorithms, such as binary exponential backoff.

References

[1] M. Cottrell, J.-C. Fort, and G. Malgouyres. Large Deviations and Rare Events in the Study of Stochastic Algorithms. *IEEE Trans. Auto. Control* **AC-28** (1983), 907–920.

[2] M. I. Freidlin and A. D. Wentzell. *Random Perturbations of Dynamical Systems.* Springer-Verlag, 1984.

[3] H. Goldstein. *Classical Mechanics.* Addison-Wesley, second edition, 1980.

[4] J. Goodman, A. G. Greenberg, N. Madras, and P. March. The Stability of Binary Exponential Backoff. *J. Assoc. Comput. Mach.* **35** (1988), 579–602.

[5] N. J. Günther and J. G. Shaw. Path Integral Evaluation of ALOHA Network Transients. *Inf. Process. Lett.* **33** (1990), 289–295.

[6] J. C. Lagarias and A. Weiss. The $3x + 1$ Problem: Two Stochastic Models. AT&T Bell Laboratories preprint, 1989.

[7] R. S. Maier. Colliding Stacks: A Large Deviations Analysis. University of Arizona preprint, 1989.

[8] R. S. Maier. The Asymptotic Evolution of Data Structures. In *Proceedings of ICCI '90: The Second International Colloquium on Computing and Information*, Niagara Falls, 1990. Springer-Verlag.

[9] W. A. Rosenkrantz and D. Towsley. On the Instability of the Slotted ALOHA Multiaccess Algorithm. *IEEE Trans. Auto. Control* **AC-28** (1983), 994–996.

[10] H. Takagi and L. Kleinrock. Throughput Analysis for Persistent CSMA Systems. *IEEE Trans. Comm.* **COM-33** (1985), 627–638.

[11] F. A. Tobagi. Multiaccess Link Control. In *Computer Network Architectures and Protocols*, edited by P. E. Green. Plenum Press, 1982.

[12] A. D. Ventcel. Rough Limit Theorems for Large Deviations for Markov Processes, I. *Theory Probab. Appl.* **21** (1976), 227–242.

[13] A. D. Ventcel. Rough Limit Theorems for Large Deviations for Markov Processes, II. *Theory Probab. Appl.* **21** (1976), 499–512.

[14] A. Weiss. A New Technique for Analysing Large Traffic Systems. *Adv. in Appl. Probab.* **18** (1986), 506–532.

CONTENTION RESOLUTION AMONG FOUR SENDERS

James E. Jantzen
Rockwell International Corporation
400 Collins Road NE
Cedar Rapids, Iowa 52498

Mark S. Andersland
Electrical and Computer Engineering Dept.
The University of Iowa
Iowa City, Iowa 52242

ABSTRACT

A number of important contention resolution and distributed ordering problems arising in time-slotted multi-user broadcast communication systems can be viewed as generic search problems in which M independent random variables, uniformly distributed in (0,1), are to be located by testing a sequence of subsets of (0,1). This paper concerns a search problem in which $M = 4$ and each test returns a 0, a 1, or an e, depending on whether the tested subset contained 0, 1, or > 1 random variables. The average number of tests required to locate the four random variables is shown to be lower bounded by 6.495 and a testing algorithm that requires, on average, 6.782 tests is constructed.

1. INTRODUCTION

In most time-slotted, multiple-access, broadcast systems, the most recent "state" of the channel can be determined by simply "listening" to the channel. If no carrier is present, no message has been sent. If multiple carriers are present, or if a transmission is garbled, multiple messages have been sent.

When a broadcast channel is not capable of supporting simultaneous transmissions, feedback of this sort is usually termed 0,1,e-feedback. In systems in which a large number of infrequently transmitting users share a common time-slotted broadcast channel, this feedback can be used to coordinate channel access. For instance, if users with messages (senders) are permitted to decide at random whether to transmit during a given slot, the feedback can be used to adjust the senders' transmission probabilities so as to reduce the number of slots that are empty or "contested"---i.e., to increase throughput.

In distributed telemetry systems, or in multi-user systems in which power or bandwidth is limited, 0,1,e-feedback can also be used by senders to reach consensus concerning the relative order of sender specific parameters---e.g., queue lengths, job priorities, sensor readings, actuator positions, and so on---without exchanging the actual values of these parameters. For example, if at the beginning of each slot each sender transmits its address when its parameter lies between certain thresholds, the feedback can be used to adjust the next slot's thresholds such that eventually the parameters are sorted by magnitude---i.e., every sender can associate every other sender's address (parameter) with a unique interval.

Regardless of whether the 0,1,e-feedback is used to coordinate channel access or to sort distributed parameters, it is desirable to identify transmission protocols that minimize the expected number of slots that are empty or "contested." Abstractly, these protocol design problems can be viewed as generic search problems in which M independent random variables, uniformly distributed in (0,1), are to be located by testing a sequence of subsets of (0,1). That is, "good" solutions to these generic search problems can be used to develop "good" solutions to both the channel access and the distributed sort problems.

Several researchers have considered generic search problems of this type [1-6]. In [2], Arrow, Pesotchinsky, and Sobel identified the optimal algorithm [1] for locating the largest of M random variables given that each test returns the number of random variables in the tested subset (0,1,...,M-feedback). In [3], Christen determined the optimal algorithm for locating three random variables given that each test returns a 0, a 1, or an e, when the tested subset contains, respectively, 0, 1, or > 1 random variables (0,1,e-feedback). Christen's algorithm requires, on average, at least 4.833 tests to resolve contention.

In this paper we consider this 0,1,e-feedback problem when M=4. We show that the average number of tests required to locate four random variables is lower bounded by 6.495 and we construct a testing algorithm that requires, on average, 6.782 tests. The lower bound is computed by a brute force approach and is not tight. However, the approach can be generalized to compute tighter bounds and bounds for 0,1,e-problems in which M>4. The algorithm has the same basic structure as the optimal algorithms for the cases M=2 [2] and M=3 [3], but it is constructed by a different approach. The algorithm is conjectured to be optimal.

The remainder of the paper is organized as follows. In Section 2, we formulate the "Four Sender Problem." In Section 3, we lower bound the expected number of tests required to resolve the senders' contention. In Section 4, we construct a contention resolution algorithm that nearly achieves the lower bound. Section 5 contains brief concluding remarks.

2. THE FOUR SENDER PROBLEM

The "Four Sender Problem" is a generic search problem of the following form:

> Four random variables, henceforth termed active elements, are distributed uniformly in a set (0,1). A sequence of subsets S_1, S_2, ... of (0,1) are tested. Each test returns a 0, a 1, or an e, depending on whether the subset contained 0, 1, or >1 active elements. The selection of the k^th^ subset to be tested is constrained to depend on: 1) the (k - 1)th subset, 2) the result of the (k - 1)th test, and 3) the number of tests that have returned a 1. The objective is to find a subset selection algorithm that minimizes the expected number of tests required to locate all four active elements.

Note that this problem is actually non-parametric in the sense that all continuous distributions can be reduced to uniform distributions on (0,1) (cf. [2], p. 403).

The connection between this search problem and the contention resolution problem can be explained as follows. Suppose that four senders have attempted to transmit during the same slot. If each sender independently chooses a number uniformly distributed in (0,1) and if each sender only transmits during the kth slot when its number lies in the kth search set, the ratio of the measure of the kth search set to the measure of the previous search set (or complement thereof) containing the kth search set corresponds to the senders' kth transmission probability. Accordingly, the expected number of tests required to locate all four active elements equals the expected number of slots required to resolve the contention among four senders.

The connection between the search problem and the distributed ordering problem is similar. Specifically, suppose that four senders' parameters are independent, identically distributed random variables that have been reduced to independent random variables uniformly distributed in (0,1) ([2], p. 403). Then if each sender only transmits during the kth slot when its parameter lies in the kth search set, the ratio of the measure of the kth search set to the measure of the previous search set (or complement thereof) containing the kth search set corresponds to the kth threshold that senders in the distributed ordering problem should compare their parameters to. In this case, the expected number of tests required to locate all four active elements equals the expected number of slots required to completely order the four parameters.

3. A LOWER BOUND

A lower bound on the expected number of tests required to locate the four active elements is developed in this section. The bound is obtained by bounding the expected number of tests required locate the active elements given more information than would normally be available to the search algorithm. The development makes use of the following lemmas.

Lemma 1: On average,

a) at least three tests are required to locate two active elements uniformly distributed in $P \subset (0,1)$.

b) at least four tests are required to locate four active elements when each is uniformly distributed in a distinct disjoint subset of $(0,1)$.

c) at least four tests are required to locate two active elements uniformly distributed in $P \subset (0,1)$ as well as a third active element uniformly distributed in $Q \subset (0,1)-P$.

d) at least 4.833 tests are required to locate three active elements uniformly distributed in $P \subset (0,1)$.

e) at least five tests are required to locate two active elements uniformly distributed in $P \subset (0,1)$ as well as a third active element uniformly distributed in $Q \subset (0,1)-P$ and a forth active element uniformly distributed in $R \subset (0,1)-P-Q$.

f) at least 5.833 tests are required to locate three active elements uniformly distributed in $P \subset (0,1)$ as well as an active element uniformly distributed in $Q \subset (0,1)-P$.

Proof: Parts a), c), and d) follow from [3] (Lem. 2 and Thm. 3). Part b) is obvious. Parts e) and f) are established using c), d), and the approach used to prove b) of Lemma 2 in [3]. ♦

Lemma 2: On average, at least 5.576 tests are required to locate two active elements uniformly distributed $P \subset (0,1)$ as well as two active elements uniformly distributed in $Q \subset (0,1)-P$.

Proof: Let m denote the minimum expected number of tests required to locate the four active elements. An initial test of a subset $Y \cup Z$ of $P \cup Q$, $Y \subset P$, $Z \subset Q$, is conducted. Since the distributions of the elements in P and Q are independent and uniform, the probability that X contains i elements and Y contains j elements is

$$\binom{2}{i} (1-y)^{2-i} y^i \binom{2}{j} (1-z)^{2-j} z^j, \tag{1}$$

where y denotes the ratio of the measure of Y to the measure of P, and z denotes the ratio of the measure of Z to the measure of Q. By hypothesis, if $(i,j) \in \{(0,0), (0,2), (2,0), (2,2)\}$, on average at least m further tests are required (two of the four disjoint subsets Y, P-Y, Z and Q-Z contain active elements). By Lemma 1 b) and c), if $(i,j) \in \{(0,1)\ (1,0),\ (1,1)\}$, on average at least 4 further tests are required (either all four disjoint subsets Y, P-Y, Z and Q-Z contain one active element or one contains three elements and another contains one element). Similarly, by Lemma 1 e), if $(i,j) \in \{(1,2), (2,1)\}$, on average at least 5 further tests are required (two of the four disjoint subsets Y, P-Y, Z and Q-Z contain one element while a third contains two elements). Consequently,

$$\begin{aligned} m \geq \min_{y \in (0,1) z \in (0,1)} \quad & [1 + m[(1-y)^2(1-z)^2 + (1-y)^2 z^2 + y^2(1-z)^2 + y^2z^2] \\ & + 4[2(1-y)^2(1-z)z + 2(1-y)y(1-z)^2 + 4(1-y)y(1-z)z] \\ & + 5[2(1-y)yz^2 + 2y^2(1-z)z]] \end{aligned} \tag{2}$$

or

$$m \geq \frac{1+4[2(1-y)^2(1-z)z + 2(1-y)y(1-z)^2 + 4(1-y)y(1-z)z] + 5[2(1-y)yz^2 + 2y^2(1-z)z]}{1-(1-y)^2(1-z)^2 - (1-y)^2z^2 - y^2(1-z)^2 - y^2z^2}. \qquad (3)$$

Using a program such as Mathematica or MathCad, it can be verified that the unique minimum of the left hand side of (3) is 5.576 (when y=z=.374). ♦

Using Lemmas 1 and 2 we can establish the following lower bound. Its proof parallels that of Lemma 2.

Theorem 1: On average, at least 6.495 tests are required to locate four active elements uniformly distributed in (0,1).

Proof: Let m be the minimum expected number of tests required to locate the four active elements. An initial test of a subset of (0,1) of measure x is conducted. Since the distributions of the four elements are independent and uniform, the tested subset contains i active elements with probability

$$\binom{4}{i} (1-x)^{4-i}x^i . \qquad (4)$$

By hypothesis, m further tests are required on average if the first subset is empty or contains four active elements. If the subset contains one active element, by Lemma 1 d) on average at least b_3 (=4.833) further tests are required. If the subset contains two active elements, Lemma 2 implies that on average at least $b_{2,2}$ (=5.576) further tests are required. When the subset contains three active elements, Lemma 1 f) implies that on average at least b_3+1 further tests are required. Accordingly,

$$m \geq \min_{x \in (0,1)} \quad [1 + m[(1-x)^4 + x^4] + b_3 4x(1-x)^3 + b_{2,2}\, 6x^2(1-x)^2 + (b_3+1)\, 4x^3(1-x)] \qquad (5)$$

or

$$m \geq \frac{1 + b_3 4x(1-x)^3 + b_{2,2}\, 6x^2(1-x)^2 + (b_3+1)4x^3(1-x)}{1-(1-x)^4 + x^4} . \qquad (6)$$

Using a program such as Mathematica or MathCad, it can be verified that the unique minimum of the left hand side of (6) is 6.495 (when x=.365). ♦

The bound in Theorem 1 is not tight because, after the first question, it is assumed that more information is provided to the search algorithm than would actually available (e.g., it is assumed that sets containing two, three, or four active elements can be distinguished). To tighten the bound this information could be withheld until after the k^{th} question. However, the proof's final step then requires the minimization of a 2^{k-1}-valued function that is tedious to derive. Clearly this same brute force approach can be applied to cases in which M>4.

4. The "Detection of Four" Algorithm

In this section we construct an algorithm for resolving contention among four senders. The construction makes use of the two interval search procedures developed by Christen [3].

Given an interval search set $S \subset (0,1)$ known to contain at least two active elements, the "Halve" procedure, returns a "current" interval set $C \subset S$ known to contain the leftmost active element, a "next" interval set $N \subset S$ known to contain the next leftmost (second smallest) active element of S, and a "rest" interval set $R \subset S$ which is the portion of S to the right of N.

The "Halve Procedure" [3]:

```
Let the current interval C=S.
1:  Test the left half T of the current interval C;
    If T contains no active elements, let C = C-T and repeat 1;
    If T contains at least two active elements, let
    R = R + (C-T), let C = T and repeat 1;
    Otherwise, let N = C-T and exit.
```

Given an interval search set $S \subset (0,1)$ known to contain exactly three active elements, the "Detection of Three" procedure, locates the three elements.

The "Detection of Three" procedure [3]:

```
Let the current interval C = S.
1:  Test a subinterval T of measure z times the measure of the current interval C,
        where z is the root of $z^3 + 3z^2 - 9z + 3 = 0$ ($\approx 0.391$).
    If T contains zero active elements, let C = C-T and repeat 1.
    If T contains a single active element, apply "Halve" to C-T, test the N returned by
        "Halve", and exit.
    Otherwise, let C = T and let R = C-T.
    Apply "Halve" to C returning N and an updated R.  Test N.
    If N contains one active element, test R and exit.
    If N contains > 1 active element, apply "Halve" to N, test the N returned by
        "Halve", and exit.
```

When the three active elements are independent and uniformly distributed in S, the "Detection of Three" procedure is the optimal 0,1,e-feedback search procedure ([3], Thm. 3).

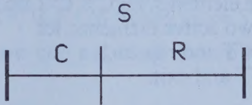
Our algorithm assumes that the interval search set $S \subset (0,1)$ is known to contain four active elements. As in Christen's procedures C, T, N, and R are subintervals of S.

The "Detection of Four" algorithm:

```
1:  Test a subinterval C of measure x times the measure of S.
    Let R = S-C.
    If C contains zero active elements, let S = R and repeat 1.
    If C contains one active element, apply the "Detection of Three" algorithm to R
        and exit.
    Otherwise, set i = 0.
2:  Test a subinterval T of measure y times the measure of C.  Let N = C-T.
    If T contains zero active elements, let C = N and repeat 2.
    If T contains >1 active elements, let C = T, let R = R + N, and repeat 2.
    Otherwise, set i = i + 1.
    Test N.
    If N contains one active element and i = 1, apply "Halve" to R, test the N
        returned by "Halve" and exit.
    If N contains one active element and i = 2, test R and exit.
    Otherwise, let C = N and repeat 2.
```

The "Detection of Four" algorithm is most easily described and analyzed from the perspective of an observer (algorithm developer) that can distinguish between search sets containing two, three, or four active elements (the algorithm itself cannot distinguish between these sets). Figure 1 details the behavior of the algorithm from this perspective. In the figure, each box corresponds to a possible algorithm state. The label next to each line exiting a state denotes the test result that would cause the algorithm to follow that path. The labels are defined as follows: 0 indicates that zero active elements were located; 1 indicates that a single active element was located; e2, e3, and e4 indicate respectively, that two, three, or four active elements were located. The x and y next to certain states denote the ratio of the measure of the test set to the measure of the current search set when in that state.

The initial state of the algorithm is START. In this state, the algorithm knows that there are four active elements in S, hence the notation "4KNOWN". The x next to the state indicates that the first test is of a subinterval C of measure x times the measure of the interval search set S. The "rest" of the search set is R.



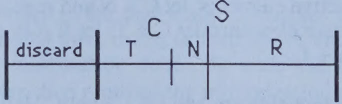
If C contains zero active elements, C is discarded and the algorithm is restarted with S set to R. That is, the algorithm remains in the START state and the next subinterval queried is of measure x times the measure of the new search set S=R.

If C contains one active element, the algorithm moves to state A. Since S contains four active elements and C contains one active element, the algorithm knows that R contains three active elements. Accordingly, it is optimal for the algorithm to apply the "Detection of Three" procedure to R to locate the remaining active elements.

If C contains two or more active elements, the state changes to "2 UNKNOWN", "3 UNKNOWN", or "4 UNKNOWN". The states are termed "unknown" because although they can be distinguished by an all-knowing algorithm developer, they can not be distinguished by the algorithm.

This is a subtle but important distinction. In past approaches, all calculations have been made from the perspective of the algorithm, that is, the constraints imposed by the 0,1,e-feedback have been accounted for at the beginning of the analysis (cf. [3]). In the present approach, complete information is assumed (i.e., 0,1,2,3,4-feedback), the 0,1,e-feedback information available to the algorithm is tracked, and the algorithm is constrained to behave identically in states that cannot be distinguished. That is, the constraints imposed by the 0,1,e-feedback are accounted for at the end of the analysis.

Formally, in each of the "unknown" states, the algorithm queries a subinterval of measure y times the measure of C, a set known to contain at least two active elements. The testing ratio y is the same for all of the unknown states because the algorithm can not distinguish among the states. In each of these unknown states there are four subintervals of the search interval to be tracked. The interval C is the "current" set known to contain two or more active elements. T is the subinterval of measure y times the measure of C, to be "tested". N is the remainder of C, the "next" set, and R is the remainder of S to the right of N, the "rest" set.



The performance of the "Detection of Four" algorithm can be evaluated recursively by computing the expected number of tests required to exit from each state.

Theorem 2: On average, the "Detection of Four" algorithm requires 6.782 tests to locate four independent active elements uniformly distributed within a search interval S.

Proof: As can be seen from Figure 1, some of the states in the algorithm are analogous and, consequently, give rise to the same equations. The seven distinct state groups are: 1) states E and L, 2) states B, D, M, and J, 3) state H, 4) states K and C, 5) state G, 6) state F, 7) state START.

For states in each of these state groups, the probabilities of the various outcomes are binomial in nature.

STATE GROUP 1: Let $u_{E,L}$ denote the expected number of tests required to locate the active elements in the states in state group 1. Since these states are entered when a test of T has located one active element, one initial test of N is performed upon entering the state. Therefore, $u_{E,L} = u_1 + 1$, where u_1 is the expected number of tests required to locate the active elements of the states in state group 1 conditioned on the outcome of the first test of T.

When zero or two active elements respond to a test of T, u_1 tests are again required by definition. When one active element responds, two more tests are required, one of N and one of R. Thus conditioning on the outcome of the first test of T, the expected number of required tests is

$$u_1 = 1 + [\tbinom{2}{0}\, y^0\, (1\text{-}y)^2 + \tbinom{2}{2}\, y^2\, (1\text{-}y)^0]\, u_1 + \tbinom{2}{1}\, y\, (1\text{-}y)\, (1+1)\ . \tag{7}$$

Solving for u_1 in Eq. (9) and adding the initial test of N, $u_{E,L}$ becomes

$$u_{E,L} = 3 + \frac{1}{2\ y\ (1\text{-}y)}\ . \tag{8}$$

STATE GROUP 2: Let $u_{B,D,M,J}$ denote the expected number of tests required to locate the active elements in the states in state group 2. Since these states are entered when the current interval is known to contain more than two active elements, the first test performed is on a new subinterval T of C.

When zero or two active elements respond, $u_{B,D,M,J}$ further tests are required by definition. When one active element responds, a single test of N locates the second active element. By Lemma 1 of [3], on average, three further tests are required to locate the remaining two elements using “Halve” and a test of the N returned by "Halve". Consequently, conditioning on the outcome of the first test, the expected number of tests required is

$$u_{B,D,M,J} = 1 + [\tbinom{2}{0}\, y^0\, (1\text{-}y)^2 + \tbinom{2}{2}\, y^2\, (1\text{-}y)^0]\, u_{B,D,M,J} + \tbinom{2}{1}\, y\, (1\text{-}y)\, (1+3)\ . \tag{9}$$

This equation reduces to

$$u_{B,D,M,J} = 4 + \frac{1}{2\ y\ (1\text{-}y)}\ . \tag{10}$$

The analysis of state groups 3 through 6 is similar to that for state groups 1 and 2.

STATE GROUP 3: Conditioned on the outcome of the first test of T, the expected number of tests required to locate the active elements in the states in state group 3 is

$$u_H = 1 + [\tbinom{2}{0}\, y^0\, (1\text{-}y)^2 + \tbinom{2}{2}\, y^2\, (1\text{-}y)^0]\, u_H + \tbinom{2}{1}\, y\, (1\text{-}y)\, (1+1)\ , \tag{11}$$

which reduces to

$$u_H = 2 + \frac{1}{2\ y\ (1\text{-}y)}\ . \tag{12}$$

STATE GROUP 4: Conditioned on the outcome of the first test, the expected number of tests required to locate the active elements in the states in state group 4 is

$$u_{K,C} = 1 + [\tbinom{3}{0}\, y^0 (1-y)^3 + \tbinom{3}{3}\, y^3 (1-y)^0]\, u_{K,C}$$

$$+ \tbinom{3}{1}\, y\, (1-y)^2\, u_{E,L} + \tbinom{3}{2}\, y^2 (1-y)\, u_{B,D,M,J}\ , \qquad (13)$$

which reduces to

$$u_{K,C} = \frac{1 + 3\ y\ (1-y)^2\ u_E}{{}_L + 3\ y^2\ (1-y) u_{B,D,M,J}, 1\ -\ [(1-y)^3 + y^3]}\ . \qquad (14)$$

STATE GROUP 5: Conditioned on the outcome of the first test, the expected number of tests required to locate the active elements in the states of state group 5 is

$$u_5 = 1 + [\tbinom{3}{0}\, y^0 (1-y)^3 + \tbinom{3}{3}\, y^3 (1-y)^0]\, u_5 + \tbinom{3}{1}\, y\, (1-y)^2 (1+3) + \tbinom{3}{2}\, y^2 (1-y)\, u_H\ . \qquad (15)$$

Solving for u_5 in Eq. (15) and adding an initial test of N (as for state group 1), u_G becomes

$$u_G = \frac{1 + 3\ y\ (1-y)^2\ 4 + 3\ y^2\ (1-y)\ u_H}{1\ -\ [(1-y)^3 + y^3]} + 1\ . \qquad (16)$$

STATE GROUP 6: Conditioned on the outcome of the first question, the expected number of tests required to locate the active elements in the states in state group 6 is

$$u_F = 1 + [\tbinom{4}{0}\, y^0 (1-y)^4 + \tbinom{4}{4}\, y^4 (1-y)^0]\, u_F + \tbinom{4}{1}\, y\, (1-y)^3 u_G$$

$$+ \tbinom{4}{2}\, y^2 (1-y)^2\, u_{B,D,M,J} + \tbinom{4}{3}\, y^3 (1-y)\, u_{K,C}\ , \qquad (17)$$

which reduces to

$$u_F = \frac{1 + 4\ y\ (1-y)^3\ u_G + 6\ y^2\ (1-y)^2\ u_{B,D,M,J} + 4\ y^3\ (1-y)\ u_{K,C}}{1\ -\ [(1-y)^4 + y^4]}\ . \qquad (18)$$

STATE GROUP 7: The expected number of tests required to locate the active elements of state START is the same as the expected number of tests required for the entire algorithm. Let u_{START} denote the expected number of required tests. When zero active elements respond to a test of T, u_{START} further tests are required by definition. When a single active element responds, application of the "Detection of Three" algorithm requires 4.833 further tests on average by Theorem 3 of [3]. When two active elements respond, $u_{B,D,M,J}$ further tests are required. When three active elements respond, $u_{K,C}$ further tests are required. When four active elements respond, u_F further tests are required. Consequently, conditioned on the outcome of the first question, the expected number of tests required by the algorithm is

$$u_{START} = 1 + \tbinom{4}{0}\, x^0 (1-x)^4\, u_{START} + \tbinom{4}{1}\, x\, (1-x)^3 (4.833) + \tbinom{4}{2}\, x^2 (1-x)^2\, u_{B,D,M,J}$$

$$+ \tbinom{4}{3}\, x^3 (1-x)\, u_{K,C} + \tbinom{4}{4}\, x^4 (1-x)^0\, u_F\ , \qquad (19)$$

which reduces to

$$u_{START} = \frac{1 + 4\ x\ (1-x)^3\ 4.833 + 6\ x^2\ (1-x)^2\ u_{B,D,M,J} + 4\ x^3\ (1-x)\ u_{K,C} + x^4\ u_F}{1\ -\ (1-x)^4}\ . \qquad (20)$$

Using a program such as Mathematica or MathCad, it can be verified that the unique minimum of (20), 6.782, is obtained when $x = 0.295$ and $y = 0.488$. ♦

5. Conclusion

This paper concerned a generic search problem in which four independent random variables, uniformly distributed in (0,1), are to be located by testing a sequence of subsets of (0,1) Assuming that each test provides 0,1,e-feedback, it was shown that, on average, at least 6.495 tests are required to locate the random variables. A testing algorithm that requires, on average, 6.782 test was also developed. The approach used derive the lower bound can be generalized to compute tighter bounds and bounds for 0,1,e-problems involving more than four senders. The approach used to construct the testing algorithm can also be generalized to larger problems. These results are of interest because, as outlined in Section 2, they can be used to develop good algorithms for resolving practical contention resolution and distributed ordering problems.

REFERENCES

[1]. V. Anantharam and P. Varaiya, "An Optimal Strategy For a Conflict Resolution Problem," Systems & Control Letters, vol. 7, no. 4, pp. 329-332, 1986.

[2]. Kenneth J. Arrow, Leon Pesotchinsky, and Milton Sobel, "On Partitioning a Sample With Binary-Type Questions in Lieu of Collecting Observations," Journal of the American Statistical Association, vol. 76, no. 374, pp. 402-409, 1981.

[3]. Claude A. Christen, "Optimal Contention Among Three Senders," IEEE Transactions on Information Theory, vol. 34, no. 4, pp. 851-857, 1988.

[4]. Mart L. Molle, "On the Capacity of Infinite Population Multiple Access Protocols," IEEE Transactions on Information Theory, vol. IT-28, no. 3, pp. 396-401, 1982.

[5]. Nicholas Pippenger, "Bounds on the Performance of Protocols for a Multiple-Access Broadcast Channel," IEEE Transactions on Information Theory, vol. IT-27, no. 2, pp. 145-151, 1981.

[6]. B. S. Tsybakov, "Resolution of a conflict of known multiplicity," Problems of Information Transmission, vol. 16, no. 2, pp. 134-144, 1980.

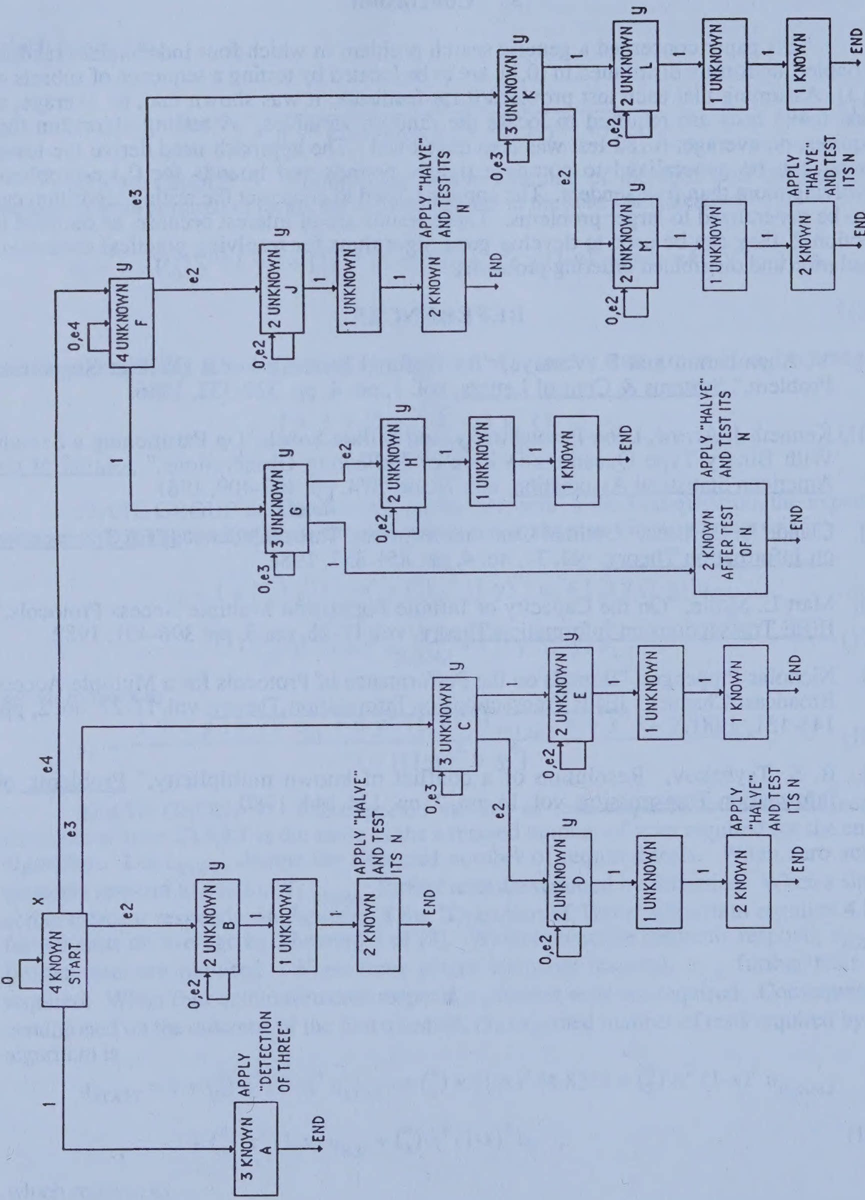

4 KNOWN START
0
x
1
3 KNOWN A
APPLY "DETECTION OF THREE"
END
e2
2 UNKNOWN B
0,e2
y
1
1 UNKNOWN
1
2 KNOWN
APPLY "HALVE" AND TEST ITS N
END
e3
e4
3 UNKNOWN C
0,e3
y
e2
2 UNKNOWN D
0,e2
y
1
1 UNKNOWN
1
2 KNOWN
APPLY "HALVE" AND TEST ITS N
END
1
2 UNKNOWN E
0,e2
y
1
1 UNKNOWN
1
1 KNOWN
END
4 UNKNOWN F
0,e4
y
1
3 UNKNOWN G
0,e3
y
e2
2 UNKNOWN H
0,e2
y
1
1 UNKNOWN
1
1 KNOWN
1
2 KNOWN AFTER TEST OF N
END
APPLY "HALVE" AND TEST ITS N
e2
2 UNKNOWN J
0,e2
1
1 UNKNOWN
1
2 KNOWN
APPLY "HALVE" AND TEST ITS N
END
e3
3 UNKNOWN K
0,e3
y
e2
2 UNKNOWN M
0,e2
y
1
1 UNKNOWN
1
2 KNOWN
APPLY "HALVE" AND TEST ITS N
END
1
2 UNKNOWN L
0,e2
y
1
1 UNKNOWN
1
1 KNOWN
END

Figure 1. Detection of Four Algorithm

FH-MFSK MULTIPLE-ACCESS COMMUNICATIONS FOR THE FACTORY ENVIRONMENTS

PARVIZ YEGANI
Department of Electrical Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794

At Allerton '89 a Frequency-Hopped M-ary Frequency-Shift-Keyed Spread-Spectrum Communication System was introduced and the design procedure for the system operating over a Factory Radio Channel was presented [1]. The performance of the system operating over the Rayleigh fading was considered. In this paper, channel probabilities for Rician and logNormal multipath fading for factory environments are formulated and numerically computed. These probabilities are used to evaluate the performance of the FH-MFSK system. The statistics of these channels were obtained from a recent channel model [2-4]. The bit error probability for the FH-MFSK can be written as

$$P_e = P_{e,0}(1 - \hat{P}) + P_{e,1}\hat{P} \tag{1}$$

$P_{e,0}$ and $P_{e,1}$ are the conditional probabilities of error for any bit interval given that there are no hits or at least one hit, respectively. These probabilities are computed from eqn. (3-A) [6]. $\hat{P}$ is the probability of a hit occurring which is fixed from bit to bit for stationary random hopping patterns. If the patterns are mutually independent and identically distributed then

$$\hat{P} = 1 - (1 - p)^{K-1} \tag{2}$$

where K is the number of simultaneous users and $p = 2^{-m_2}$. As shown in the Appendix of Ref. [6] the channel probabilities (P_D and P_F) along with the system parameters are needed in order to compute the bit error probability.

The probability of deletion for a binary fading channel is given by

$$P_D = \int_0^\infty P_{D/\gamma}\, p(\gamma)\, d\gamma \tag{3}$$

where γ is the average normalized signal-to-ratio and p_γ for the AWGN is given by

$$P_{D/\gamma} = \int_0^\beta \exp\left(-\frac{2\gamma + r^2}{2}\right) I_0((2\gamma)^{1/2}r)\, r\, dr. \tag{4}$$

where β is the normalized receiver threshold. For a Rayleigh fading we have

$$p(\gamma) = 1/\bar{\gamma}\exp(-\gamma/\bar{\gamma}) \tag{5}$$

and for a Rician channel

$$p(\gamma) = 1/\bar{\gamma}\exp\left[-\frac{(\gamma+A^2)}{\bar{\gamma}}\right] I_0\left(\frac{2A\gamma^{1/2}}{\bar{\gamma}}\right) \tag{6}$$

and logNormal

$$p(\gamma) = 1/(4\pi\bar{\gamma})^{1/2}\,\gamma\exp\left[-\frac{(1/2\ln\gamma-\mu)^2}{\bar{\gamma}}\right] \tag{7}$$

Probability density function of the power signal, (6) or (7), can be derived from the pdf of the envelope fading. For the Rayleigh fading channel these probabilities for On-Off Keying (OOK) modulation scheme have closed form expressions which are given by [13]

$$P_F = \exp\left(-\frac{\beta^2}{2}\right), \quad P_D = 1 - \exp\left[-\frac{\beta^2}{2(1+\bar{\gamma})}\right] \tag{8}$$

However, for Rician and logNormal channels the closed form solutions do not exist.

By incorporating (4) ,(5), and (6) in (3) we get

$$P_D = \int_0^\infty \int_0^\beta f(x,y)\, I_o(g(x,y))\, I_o(h(x,y))\, dx\, dy \qquad (9)$$

For the Rician channel we have

$$f(x,y) = 1/\bar{\gamma}\, y \exp\left(- \frac{\bar{\gamma}x + 0.5\bar{\gamma}y^2 + A^2}{\gamma}\right) \qquad (10)$$

where A is the unfaded component of the received signal.

$$g(x,y) = 2A/\bar{\gamma}x^{1/2}, \qquad h(x,y) = (2x)^{1/2}y \qquad (11)$$

and for the logNormal channel

$$f(x,y) = 1/(4\pi\bar{\gamma})^{1/2}\, y/x \exp\left(- \frac{\bar{\gamma}x + 0.5\bar{\gamma}y^2 + (1/2\ln x - \mu)^2}{\gamma}\right), \qquad g(x,y) = 1 \qquad (12)$$

and h(x,y) for logNormal channel also follows (11). To carry out the computations for P_D the two-dimensional integral in (10) can be handled by using the quadrature rule and IMSL Library routines. The probability of false alarm P_F can easily be obtained from (8) for all of the fading channels.

The error performance of the FH-MFSK system over a Rayleigh , Rician, and logNormal fading channels are shown in Figs. 1-2. As the figures indicate by choosing an appropriate signal to noise ratio (γ) at a fixed receiver threshold (β) a large number of users (K) can communicate within the system maintaining a prespecified Bit Error Rate (BER). The results below show that with our system a higher number of users can communicate under the specific conditions than can with more conventional systems.

For several SNRs performance of the system has been evaluated and illustrated graphically. For example, as Fig. 1 illustrates for a threshold of 3.8 and SNR of 30dB with 8-ary FSK modulation potentially more than 100 user can simultaneously transmits their data with an BER 10^{-5} or better over a Rayleigh channel. Under the logNormal fading with a mean of - 2 dB (relative to the peak value of the received signal) the system can accommodate around 100 users only with an BER of about 10^{-3}. The Rician channel with K = - 10 dB is slightly better than Rayleigh at 30 dB receiver threshold under heavy traffic. However, for light traffic (less than 50 simultaneous users) the Rician channel shows a better performance than the Rayleigh channel.

The top (dashed) curves of each plot show the error performance of the system reported in [5]. It is noticed that for any level of receiver threshold this system outperforms the one reported in [5]. For example, at a 20 dB receiver threshold less than 10 user can communicate simultaneously in system [5] with an BER of 10^{-3} over Rayleigh or Rician channel. Compare this with 100 users which can communicate successfully under the same conditions in our system (see Fig. 2). However, the degree of improvement over logNormal channel is not as good as that of other channels.

REFERENCES

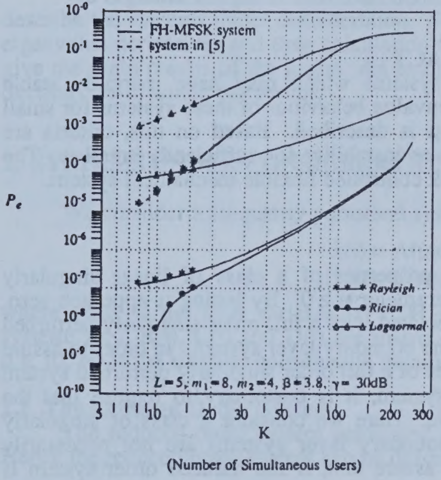
[1] P. Yegani and C. D. McGillem, "Performance of Noncoherent Frequency-Hopped, M-ary Frequency-Shift-Keying (FH-MFSK) Communcations over a Factory Radio Channel," Proc. of the 27th Annual Allerton Conference on Communications, Control, and Computing, Urbana-Champaign, IL, Sept. 1989, pp. 15-24.

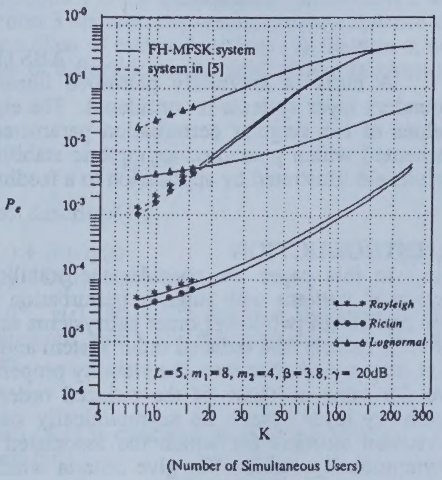
[2] P. Yegani and C. D. McGillem, "A Statistical Model for the Line-Of-Site (LOS) Factory Radio Channel," Proc. 39th IEEE Vehicular Technology Conference, San Francisco, May 1989, pp. 496-503.

[3] P. Yegani and C. D. McGillem, "A Statistical Model for Obstructed Path Factory Radio Channel," Proc. IEEE Global Communications Conference, Nov. 1989, Session 38.2, pp. 1351-1355, Dallas, TX.

[5] D. J. Goodman, P. S. Henry, and V. K. Prabhu, "Frequency-Hopped Multilevel FSK for Mobile Radio," B.S.T.J., Vol. 59, No. 7, Sept. 1980, pp. 1257-75.

[6] P. Yegani and C. D. McGillem, "Frequency-Hopped M-ary Frequency-Shift-Keying System Performance in a Factory Environment," submitted to IEEE Trans. Veh. Tech..





ON THE STABILITY OF SINGULARLY PERTURBED LINEAR SYSTEMS WHICH HAVE MARGINLY STABLE BOUNDARY LAYER SYSTEMS[†]

M. Corless , D. Da
School of Aeronautics & Astronautics
Purdue University
West Lafayette, Indiana 47907
USA

ABSTRACT

A class of singularly perturbed linear systems which may have marginly stable boundary layer systems is considered. The eigenvalue behaviour of these systems for small values of the singular perturbation parameter μ is described. Based on this, criteria are presented which guarantee asymptotic stability or instability for sufficiently small μ. The results are illustrated by application to a feedback controlled flexible mechanical system.

1. INTRODUCTION

In this paper we consider the stability properties of a class of linear singularly perturbed systems with singular perturbation parameter $\mu \geq 0$. By letting μ approach zero, one can obtain two lower order subsystems associated with a full order singularly perturbed system, namely, the reduced order system and the boundary layer system. In order to assure that, as μ approaches zero, the stability properties of a full order singularly perturbed system are the same as those of the reduced order system, it is customary to require that the boundary layer system be asymptotically stable. Here we consider a class of singularly perturbed systems for which the associated boundary layer systems are not necessarily asymptotically stable. We give criteria which assure that, if the reduced order system is asymptotically stable, the full order singularly perturbed system is asymptotically stable or unstable for sufficiently small μ.

There is some earlier work on this issue. In [2], a special structure on the boundary layer system is required; those systems turn out to be special cases of the systems considered here. In [5], by using a Lyapunov approach, a criterion is given for singularly perturbed systems whose boundary layer systems are described by vector second order differential equations. The results in this paper are more general.

The approach we use in this paper is to describe the eigenvalue changes of a singularly perturbed linear system for small values of the perturbation parameter μ. Then we present stability and instability criteria for small μ. Before that a result on the eigenvalue derivatives of a class of matrix functions is presented, which is not only useful for our main results, but of importance in itself. Some earlier work has been done on the problem of eigenvalue and eigenvector derivatives; see [8,9,13,17]. Most of this work has some nondefective assumptions on the matrix function. In our paper the matrix function is not necessarily nondefective.

The motivation for the study of the stability properties of a singularly perturbed system is based on the fact that one must neglect some dynamics to obtain a suitable order model when modeling a real dynamic system. The question that arises is whether this model can be used to describe the behavior of the "real" system when the neglected part is very "small". One way to answer this question is to describe the "real" system as a singularly perturbed system for which the associated reduced order system describes the chosen model. Hence the singular perturbation parameter μ represents the effect of the neglected dynamics.

[†] Based on research supported by the U.S. National Science Foundation under grants MSM-87-06927 and MSS-90-57079

The model is considered to be suitable if the full order singularly perturbed system has the same qualitative properties as that of the reduced order system for sufficiently small μ.

Flexible mechanical systems are good examples to illustrate this; see [1,3,4,7,14,15]. One can describe a flexible mechanical system as a singularly perturbed system for which the associated reduced order system describes a rigid model (it is also possible to include some flexible modes in the reduced order system, see [3]). Suppose a controller is designed to stabilize a rigid model. The results in this paper present conditions which assure that the full order system is asymptotically stable or unstable when this system is sufficiently stiff and the associated boundary layer system is not necessarily asymptotically stable.

We organize the paper into 5 sections. Section 1 is the Introduction. In Section 2 we describe the systems under consideration. Section 3 contains the results on the existence of eigenvalue derivatives and their calculation for a class of matrix functions. In Section 4, we give the main results of the paper. An example of a flexible mechanical system is given in Section 5 to illustrate our results.

2. SYSTEMS UNDER CONSIDERATION

We consider a singularly perturbed system described by

$$\dot{x}(t) = A(\mu)x(t) + B(\mu)y(t) \tag{2.1a}$$
$$\mu\dot{y}(t) = C(\mu)x(t) + D(\mu)y(t) \tag{2.1b}$$

where $t \in \mathbb{R}$ is the time variable; $x(\cdot) : \mathbb{R} \to \mathbb{C}^n$, $y(\cdot) : \mathbb{R} \to \mathbb{C}^m$ describe the state of the system; $A(\cdot)$, $B(\cdot)$, $C(\cdot)$ and $D(\cdot)$ are matrix functions which are differentiable at zero; and $\mu \in [0, \infty)$ is the singular perturbation parameter.

2.1 The Reduced Order System

The reduced order system associated with (2.1) is obtained by letting $\mu = 0$ in (2.1). Hence the reduced order system can be described by

$$\dot{x} = A(0)x + B(0)y \tag{2.2a}$$
$$0 = C(0)x + D(0)y\,. \tag{2.2b}$$

In order to obtain a unique reduced order system for (2.1), we make the following assumption.

Assumption A1. D(0) *is nonsingular.*

The unique reduced order system guaranteed by Assumption A1 is given by

$$\dot{x} = \bar{A}x \tag{2.3}$$
$$y = Hx \tag{2.4}$$

where

$$\bar{A} \triangleq A(0) - B(0)D(0)^{-1}C(0) \tag{2.5}$$
$$H \triangleq -D(0)^{-1}C(0)\,. \tag{2.6}$$

2.2 The Boundary Layer System

The boundary layer system associated with (2.1) is obtained by first introducing a new time variable

$$\tau \triangleq \mu^{-1}t \tag{2.7}$$

and considering the behavior of the resulting system for $\mu = 0$. Defining

$$\xi(\tau) \triangleq x(\mu\tau) = x(t) \tag{2.8a}$$
$$\eta(\tau) \triangleq y(\mu\tau) = y(t) \tag{2.8b}$$

and utilizing (2.1), we obtain

$$\xi' = \mu[A(\mu)\xi + B(\mu)\eta] \tag{2.9a}$$
$$\eta' = C(\mu)\xi + D(\mu)\eta \,. \tag{2.9b}$$

Letting $\mu = 0$ in (2.9) yields

$$\xi' = 0 \tag{2.10}$$
$$\eta' = C(0)\xi + D(0)\eta \,. \tag{2.11}$$

Equation (2.10) implies that η is constant; i.e.

$$\xi(\tau) = \xi(0) \qquad \forall \tau \in \mathbb{R} \,.$$

Since D(0) is invertible, (2.11) has a unique equilibrium solution given by

$$\eta(\tau) = H\xi(0) \tag{2.12}$$

where H is given by (2.6).

As mentioned in Section 1, in most of the literature dealing with the stability of linear singularly perturbed systems, it is required that the boundary layer system (2.11) be asymptotically stable about its equilibrium solution given by (2.12); i.e., the eigenvalues of D(0) should be in open left half of the complex plane. There are some exceptions [2,5]; there some specific structural requirements are placed on the matrix D(0). We consider here systems for which the boundary layer systems are not necessarily asymptotically stable.

3. EIGENVALUE DERIVATIVES FOR A CLASS OF MATRIX-VALUED FUNCTIONS.

Before we present the results for the systems described in the previous section, we consider eigenvalue derivatives for a class of matrix-valued functions $F(\cdot)$: $\mathbb{R} \to \mathbb{C}^{n \times n}$ which are differentiable at zero. For a matrix function which is differentiable at zero, one might expect that its eigenvalues and eigenvectors are differentiable at zero. Unfortunately, this is not always true; see Example 3.1. Since the differentiability of eigenvalues is of importance for our results and many other applications, we present a theorem which characterizes a class of eigenvalue derivatives of a matrix-valued function $F(\cdot)$.

There is some earlier work on this issue. In [8], a proof is given for the existence of eigenvalue and eigenvector derivatives of matrix functions which are nondefective for each $\mu \in \mathbb{R}$. In [13], some results are given for the case in which the nominal matrix, F(0), is nondefective. Reference [9] has similar results. In this paper, we present an existence result and a calculation method for the eigenvalue derivatives corresponding to the nondefective eigenvalues of F(0) (see Definition 3.1); we do not require the matrix F(0) to be nondefective. A similar problem has been considered in [17]. Their results on this issue are more conservative than those presented here.

First a definition is given for the class of eigenvalues in which we are interested.

Definition 3.1. *An eigenvalue λ of a matrix $M \in \mathbb{C}^{n \times n}$ is a nondefective eigenvalue of M iff its geometric and algebraic multiplicities are equal.*

Theorem 3.1. *Suppose $F(\cdot) : \mathbb{R} \to \mathbb{C}^{n \times n}$ is differentiable at zero and $\lambda(\cdot) : \mathbb{R} \to \mathbb{C}$ is an eigenvalue function for $F(\cdot)$ such that $\lambda(\cdot)$ is continuous at zero and $\lambda(0)$ is a nondefective eigenvalue of F(0) with multiplicity m_o. Then the following hold.*

(i) There exist matrices $R = [r_1 r_2 \cdots r_{m_o}]$ and $L = [l_1 l_2 \cdots l_{m_o}]^*$ such that $\{r_1, r_2, \ldots, r_{m_o}\}$ and $\{l_1, l_2, \ldots, l_{m_o}\}$ are bases of the right and left eigenspaces,[†] *respectively, of* F(0)

[†] Suppose $\lambda \in \mathbb{C}$ is an eigenvalue of a matrix $M \in \mathbb{C}^{n \times n}$. The right eigenspace of the matrix M associated with λ is the set of all vectors $x \in \mathbb{C}^n$ satisfying $Mx = \lambda x$. The left eigenspace of M associated with λ is the set of all vectors $y \in \mathbb{C}^n$ satisfying $y^* M = \lambda y^*$.

which are associated with the eigenvalue $\lambda(0)$*, and*

$$LR = I_{m_o} \tag{3.1}$$

where $I_{m_o} \in \mathbb{R}^{m_o \times m_o}$ *is an identity matrix.*

(ii) $\lambda(\cdot)$ *is differentiable at zero and* $\lambda'(0)$ *is an eigenvalue of the matrix* $LF'(0)R$*, where L and R are any matrices which assure (i).*

Proof. Reference [6] contains a proof.

Two simple examples are given in the following to illustrate the above theorem.

Example 3.1. Consider a matrix function given by

$$F(\mu) = \begin{bmatrix} 0 & 1 \\ \mu & 0 \end{bmatrix}. \tag{3.2}$$

The only eigenvalue of F(0) is $\lambda_0 = 0$. Since the geometric multiplicity of λ_0 is 1 while its algebraic multiplicity is 2, λ_0 is not a nondefective eigenvalue of F(0). The eigenvalues of $F(\mu)$ are

$$\lambda_{1,2}(\mu) = \pm\mu^{1/2} \tag{3.3}$$

which are not differentiable at zero.

Example 3.2. Consider a matrix function given by

$$F(\mu) = \begin{bmatrix} 0 & \mu \\ \mu & 0 \end{bmatrix}. \tag{3.4}$$

The only eigenvalue of F(0) is $\lambda_o = 0$ and λ_o is a nondefective eigenvalue. Using Theorem 3.1, we choose

$$R = L = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.5}$$

Since

$$LF'(0)R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{3.6}$$

the eigenvalues of (3.4) are differentiable at zero and the eigenvalue derivatives $\lambda'(0)$ are the eigenvalues of $LF'(0)R$. Hence

$$\lambda'_{1,2}(0) = \pm 1\,. \tag{3.7}$$

This agrees with the fact that the eigenvalues of (3.4) are

$$\lambda_{1,2}(\mu) = \pm\mu\,. \tag{3.8}$$

4. EIGENVALUE ANALYSIS AND STABILITY PROPERTIES OF A CLASS OF SINGULARLY PERTURBED LINEAR SYSTEMS

We consider a singularly perturbed system described by (2.1) and satisfying Assumption A1. Using the result of the previous section, we can describe how the eigenvalues of (2.1)[†] change near $\mu = 0$. We let the eigenvalues of the reduced order system

---

[†] Here, an eigenvalue of a linear system $\dot{\phi} = M\phi$ means an eigenvalue of the matrix M.

(2.3), (2.5) associated with (2.1) be

$$\{\sigma_1, \sigma_2, ..., \sigma_n\} \tag{4.1}$$

and the eigenvalues of the boundary layer system (2.11), i.e., the eigenvalues of D(0), be

$$\{\sigma_{n+1}, \sigma_{n+2}, ..., \sigma_{n+m}\} . \tag{4.2}$$

Without loss of generality, for any eigenvalue σ_i in (4.1) or (4.2) with algebraic multiplicity $m_i \geq 1$, we suppose

$$\sigma_i = \sigma_{i+1} = \cdots = \sigma_{i+m_i-1} . \tag{4.3}$$

Suppose an eigenvalue of D(0), say σ_i, is nondefective with multiplicity m_i. Let $R_i \in \mathbb{C}^{m \times m_i}$ and $L_i \in \mathbb{C}^{m_i \times m}$ be such that the columns of R_i and L_i^* span the right and left eigenspaces, respectively, associated with σ_i and

$$L_i R_i = I_{m_i} . \tag{4.4}$$

By Theorem 3.1 (i), such matrices exist. We let

$$\tilde{D} \triangleq D'(0) + D(0)^{-1} C(0) B(0) \tag{4.5}$$

and

$$\tilde{D}_i \triangleq L_i \tilde{D} R_i , \tag{4.6}$$

and the eigenvalues of $\tilde{D}_i$ be

$$\{d_i, d_{i+1}, ..., d_{i+m_i-1}\} . \tag{4.7}$$

The following theorem describes the eigenvalue behavior of system (2.1) near $\mu = 0$.

Theorem 4.1. *Consider a singularly perturbed system described by (2.1) and satisfying Assumption A1. Then, for (2.1), there exist n+m eigenvalue functions* $\lambda_i(\cdot)$, $i = 1, 2, ..., n+m$, *such that the following hold.*

(i)
$$\lim_{\mu \to 0} \lambda_i(\mu) = \sigma_i \qquad i = 1, 2, ..., n \tag{4.8}$$
$$\lim_{\mu \to 0} \mu\lambda_i(\mu) = \sigma_i \qquad i = n+1, n+2, ..., n+m \tag{4.9}$$

(ii) Suppose $\sigma_i = \sigma_{i+1} = \cdots = \sigma_{i+m_i-1}$ *is a nondefective eigenvalue of* D(0) *with multiplicity* m_i. *Then*

$$\lim_{\mu \to 0} [\lambda_j(\mu) - \mu^{-1}\sigma_j - d_j] = 0 \qquad j = i, i+1, \cdots i+m_i-1 , \tag{4.10}$$

where d_j *is an eigenvalue of* $\tilde{D}_i$ *defined by (4.5), (4.6).*

Proof. This Theorem is proven using Theorem 3.1. The details are contained in [6].

As an application of Theorem 4.1, we consider the stability properties of a singularly perturbed system described by (2.1) and satisfying Assumption A1. We make the following assumptions on (2.1).

Assumption A2 *The eigenvalues of* $A(0) - B(0)D(0)^{-1}C(0)$ *given by (4.1) have negative real parts; i.e.,*

$$Re(\sigma_i) < 0 \qquad i = 1, 2, ..., n . \tag{4.11}$$

Assumption A2 guarantees that the reduced order system associated with (2.1) is asymptotically stable.

Assumption A3. $D(0)$ *is stable*[†].

Assumption A3 assures that the boundary layer system associated with (2.1) is at least marginly stable and that there exists a nonsingular matrix $T \in \mathbb{C}^{m \times m}$ such that

$$T^{-1}D(0)T = \begin{bmatrix} \bar{D} & 0 & & 0 \\ 0 & j\omega_1 I_{m_1} & & 0 \\ & & \ddots & \\ 0 & 0 & & j\omega_l I_{m_l} \end{bmatrix} \tag{4.12}$$

where $j \triangleq \sqrt{-1}$; $\bar{D} \in \mathbb{C}^{m_o \times m_o}$ is an asymptotically stable matrix[‡]; $I_{m_i} \in \mathbb{R}^{m_i \times m_i}$, $i = 1, 2, \cdots l$ are identity matrices; $j\omega_i$, $i = 1, 2, \cdots l$, are the imaginary eigenvalues of $D(0)$ with $\omega_i \neq \omega_j$ for $i \neq j$ and $m_i \geq 0$, $i = 0, 1, \cdots l$, are integers with

$$\sum_{i=0}^{l} m_i = m\,. \tag{4.13}$$

Let

$$T^{-1}[D'(0) + D(0)^{-1}C(0)B(0)]T = \begin{bmatrix} * & * & \cdots & * \\ * & \tilde{D}_{11} & \cdots & * \\ \vdots & \vdots & \ddots & \\ * & * & & \tilde{D}_{ll} \end{bmatrix} \tag{4.14}$$

where $\tilde{D}_{ii} \in \mathbb{C}^{m_i \times m_i}$. Then the following corollary yields a stability criterion for system (2.1).

Corollary 1. *Consider a singularly perturbed system described by (2.1) and satisfying Assumptions A1-A3. Suppose all the eigenvalues of the matrices* $\tilde{D}_{ii}$ $i = 1, 2, \cdots l$, *have negative real parts, where* $\tilde{D}_{ii}$, $i = 1, 2, \cdots l$ *are given by (4.14). Then there exists a* $\mu^* > 0$ *such that for all* $\mu \in (0, \mu^*)$, *(2.1) is asymptotically stable.*

The following corollary yields an instability criterion.

Corollary 2. *Consider a singularly perturbed system described by (2.1) and satisfying Assumptions A1-A3. Suppose that at least one of the eigenvalues of* $\tilde{D}_{ii}$, $i = 1, 2, \cdots l$, *has a positive real part. Then there exists a* $\mu^* > 0$, *such that for all* $\mu \in (0, \mu^*)$, *(2.1) is unstable.*

Proof (of Corollary 1 and 2). Reference [6] contains a proof.

[†] We define a square matrix M to be stable if the system described by $\dot{x} = Mx$ is stable. This is equivalent to the requirement that all the eigenvalues of M be in the closed left half of the complex plane and the eigenvalues on the imaginary axis are nondefective.

[‡] We define a square matrix M to be asymptotically stable if the system described by $\dot{x} = Mx$ is asymptotically stable. This is equivalent to the requirement that all the eigenvalues of M be in the open left half of the complex plane.

5. EXAMPLE: A FLEXIBLE MECHANICAL SYSTEM

In this section we apply the results of the previous section to the flexible dual rudder steering mechanism shown in Figure 1. Each part of the system is constrained to rotate about a line parallel to the inertially fixed direction ê. P_1 is a rotor with moment of inertia J_1 about L_1 and its angular displacement is q_1. P_2 is a gear assembly composed of three identical gears, each of which has moment of inertia J_2 about its axis of rotation. We let q_2 be the angular displacements of the gears as shown in Figure 1. P_1 and P_2 are connected by a massless shaft B_1. Two rudders P_3 and P_4 with moments of inertia J_3 and J_4 about L_2 and L_3 are connected to P_2 by two massless shafts B_2 and B_3, respectively; q_3 and q_4 are the angular displacements of P_3 and P_4, respectively. Suppose an angular displacement sensor is placed on P_3 and an angular rate sensor is placed on P_4. Any linear memoryless output feedback controller has the form

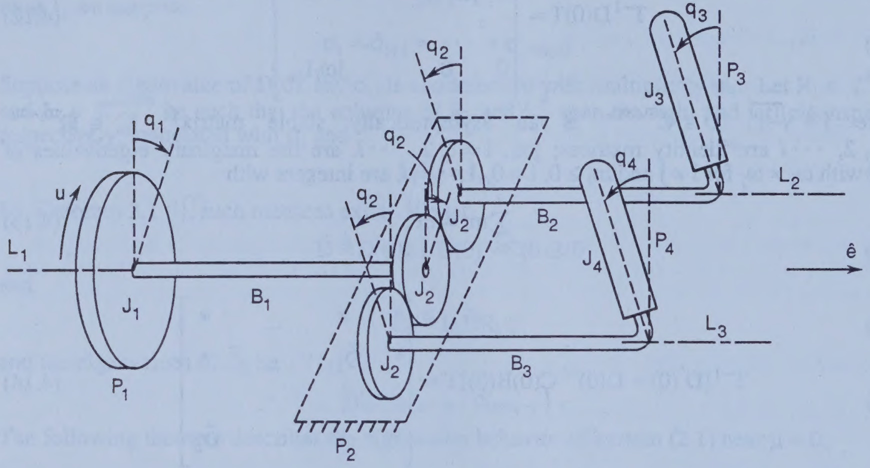


Figure 1

$$u = -l_1 q_3 - l_2 \dot{q}_4 \tag{5.1}$$

The problem we consider here is as follows. Design a controller which stabilizes a rigid model of the system; i.e., a model in which B_i, i = 1, 2, 3 are considered rigid. Then determine conditions under which this controller also stabilizes a flexible model provided the flexible elements are sufficiently stiff.

The rigid model of the system is obtained by considering B_i, i = 1, 2, 3 rigid. In this case, the system is described by

$$J_T \ddot{\bar{q}} = u , \tag{5.2a}$$

where $\bar{q} \triangleq q_1 = q_2 = q_3 = q_4$, and

$$J_T \triangleq J_1 + 3J_2 + J_3 + J_4 . \tag{5.2b}$$

The feedback controlled rigid model is

$$J_T \ddot{\bar{q}} = -l_1 \bar{q} - l_2 \dot{\bar{q}} . \tag{5.3}$$

Hence for any $l_1, l_2 > 0$, (5.3) is asymptotically stable.

Suppose now the B_i, $i = 1, 2, 3$ are considered flexible. We model each B_i, $i = 1, 2, 3$, as a flexible shaft of linear torsional stiffness coefficient $k > 0$ and linear damping coefficient $c \geq 0$. Thus the feedback controlled flexible model of the system can be described by the following equations

$$\begin{aligned} J_1\ddot{q}_1 &= k(q_2 - q_1) + c(\dot{q}_2 - \dot{q}_1) - l_1 q_3 - l_2 \dot{q}_4 && (5.4a) \\ 3J_2\ddot{q}_2 &= k(q_1 - 3q_2 + q_3 + q_4) + c(\dot{q}_1 - 3\dot{q}_2 + \dot{q}_3 + \dot{q}_4) && (5.4b) \\ J_3\ddot{q}_3 &= -k(q_3 - q_2) - c(\dot{q}_3 - \dot{q}_2) && (5.4c) \\ J_4\ddot{q}_4 &= -k(q_4 - q_2) - c(\dot{q}_4 - \dot{q}_2)\,. && (5.4d) \end{aligned}$$

Suppose the system parameters are

$$J_1 = J_3 = J_4 = 1\,,\ J_2 = \frac{1}{3}$$

and we let

$$\mu \triangleq k^{-1/2} \quad ; \quad x_1 \triangleq q_2 \quad ; \quad x_2 \triangleq \dot{q}_2 \ ;$$
$$y_1 \triangleq k(q_1 - q_2) \quad ; \quad y_2 \triangleq k(q_3 - q_2) \quad ; \quad y_3 \triangleq k(q_4 - q_2) \ ;$$
$$y_4 \triangleq k^{1/2}(\dot{q}_1 - \dot{q}_2) \ ; \ y_5 \triangleq k^{1/2}(\dot{q}_3 - \dot{q}_2) \ ; \ y_6 \triangleq k^{1/2}(\dot{q}_4 - \dot{q}_2)\,.$$

Then system (5.4) can be described by the following singularly perturbed system

$$\begin{aligned} \dot{x} &= A(\mu)x + B(\mu)y && (5.5a) \\ \mu\dot{y} &= C(\mu)x + D(\mu)y && (5.5b) \end{aligned}$$

where

$$A(\mu) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \ B(\mu) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & \mu c & \mu c & \mu c \end{bmatrix} \qquad (5.5c)$$

$$D(\mu) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -2 & -1-\mu^2 l_1 & -1 & -2\mu c & -\mu c & -\mu(c+l_2) \\ -1 & -2 & -1 & -\mu c & -2\mu c & -\mu c \\ -1 & -1 & -2 & -\mu c & -\mu c & -2\mu c \end{bmatrix}, \ C(\mu) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -l_1 & -l_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \qquad (5.5d)$$

The reduced order system for (5.5) is

$$\dot{x} = \bar{A}x \qquad (5.6)$$

where

$$\bar{A} = A(0) - B(0)D(0)^{-1}C(0) = \begin{bmatrix} 0 & 1 \\ -\frac{1}{4}l_1 & -\frac{1}{4}l_2 \end{bmatrix} \qquad (5.7)$$

which is exactly the feedback controlled rigid model (5.3). Hence (5.6) is asymptotically stable for any $l_1, l_2 > 0$.

It can readily be shown that there exists $T \in \mathcal{C}^{6\times 6}$, $\det(T) \neq 0$, such that

$$T^{-1}D(0)T = \mathrm{diag}\,[2j, -2j, j, j, -j, -j] \qquad (5.8)$$

Hence the boundary layer system for (5.5) is just marginally stable. We use Corollaries 1 and 2 to determine the stability properties of (5.5) for sufficiently small μ. Since

$$D'(0) + D(0)^{-1}D(0)B(0) = cD_1 + l_2 D_2$$

where

$$D_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & -1 & -1 \\ 0 & 0 & 0 & -1 & -2 & -1 \\ 0 & 0 & 0 & -1 & -1 & -2 \end{bmatrix}, \qquad D_2 = \frac{1}{4}\begin{bmatrix} 3 & 3 & 3 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

it can readily be shown that

$$-T^{-1}[D'(0) + D(0)^{-1}C(0)B(0)]T = \begin{bmatrix} d_1 & * & * & * \\ * & d_1 & * & * \\ * & * & \overline{D} & * \\ * & * & * & \overline{D} \end{bmatrix}$$

where

$$d_1 = 2c + 0.0417\, l_2\,, \tag{5.9}$$

$$\overline{D} = \begin{bmatrix} 0.5c - l_2(0.1420 - 0.0564j) & l_2(0.0101 + 0.2887j) \\ l_2(0.0237 - 0.0223j) & 0.5c - l_2(0.0246 + 0.0564j) \end{bmatrix}$$

and

$$\sigma(\overline{D}) = \{0.5c,\ 0.5c - l_2/6\}\,. \tag{5.10}$$

Using Corollary 1, the system is asymptotically stable for sufficiently small μ if

$$\mathrm{Re}(d_1) > 0 \quad \text{and} \quad \mathrm{Re}(\lambda) > 0 \ \forall\ \lambda \in \sigma(\overline{D}).$$

Hence, by considering (5.9) (5.10), the condition

$$3c > l_2 \tag{5.11}$$

guarantees asymptotic stability of (5.5) for sufficiently small μ.

Similarly, using Corollary 2, the system is unstable for sufficiently small μ if

$$3c < l_2\,. \tag{5.12}$$

CONCLUSIONS

An existence result and a calculation method for the eigenvalue derivatives of a class of matrix functions are presented. Then, the eigenvalue behaviour of a class of singularly perturbed linear systems for small μ is presented; from this, stability and instability criteria are obtained. The results can be used for a singularly perturbed system which has a marginly stable boundary layer system (which is customarily required to be asymptotically stable). An example of a flexible mechanical system is given to illustrate the results.

REFERENCES

[1] Chernous'ko, F. L., and Shamaev, A. S., *Asymptotic Behavior of Singular Perturbations in the Problem of Dynamics of a Rigid Body with Elastic and Dissipative Elements,* Mechanics of Solids, Vol. 18, pp. 31-41 (1983).

[2] Chow, J. H., Allemong, J. J., and Kokotovic, P. V., *Singular Perturbation Analysis of Systems with Sustained High Frequency Oscillations,* Automatica, Vol. 14, pp. 271-279 (1978).

[3] Corless, M., *Stability Robustness of Linear Feedback Controlled Mechanical Systems in the Presence of Unmodelled Flexibilities,* Proceedings of the 17th Conference on Decision and Control, Austin, Texas (1988).

[4] Corless, M., *Controllers Which Guarantee Robustness with Respect to Unmodelled Flexibilities for a Class of Uncertain Mechanical Systems,* International Journal of Adaptive Control and Signal Processing, to appear.

[5] Corless, M., *Asymptotic Stability of Singularly Perturbed Systems Which Have Marginally Stable Boundary Layer Systems,* Proceedings of the 3rd USC Workshop on Control Mechanics, Los Angeles, California (1990).

[6] Corless, M., and Da, D., *Stability Properties of Singularly Perturbed Linear Systems with Marginly Stable Boundary Layer Systems,* in preparation.

[7] Ficola, A., Marino, R., and Nicosia, S., *A Singular Perturbation Approach to the Dynamic Control of Elastic Robots,* Proceedings of the 21st Annual Allerton Conference on Communication, Control, and Computing, University of Illinois (1983).

[8] Juang, J., Ghaemmaghami, P., and Lim, K. B., *Eigenvalue and Eigenvector Derivatives of a Nondefective Matrix,* Journal Guidance, Vol. 12, pp. 480-486 (1989).

[9] Kato, T., *A Short Introduction to Perturbation Theory for Linear Operators,* Springer-Verlag (1982).

[10] Kokotovic, P. V., Khalil, H. K., and O'Reilly, J., *Singular Perturbation Methods in Control: Analysis and Design,* Academic Press, London (1986).

[11] Kokotovic, P. V., and Khalil, H. K. (editors), *Singular Perturbations in Systems and Control,* IEEE Press (1986).

[12] Kokotovic, P. V., *Applications of Singular Perturbation Techniques to Control Problems,* SIAM Review, Vol. 26, pp. 396-416 (1984).

[13] Lancaster, P., *On Eigenvalues of Matrices Dependent on a Parameter,* Numerische Mathematik, Vol. 6, pp. 377-387 (1964).

[14] Marino, R., and Nicosia, S., *On the Feedback Control of Industrial Robots with Elastic Joints: A Singular Perturbation Approach,* University of Rome, R-84.01 (1984).

[15] Spong, M. W., Khorasani, K., and Kokotovic, P. V., *An Integral Manifold Approach to the Feedback Control of Flexible Joint Robots,* IEEE Journal of Robotics and Automation, Vol. RA-3, pp. 291-300 (1987).

[16] Vidyasagar, M., *Nonlinear System Analysis,* Prentice-Hall Inc. (1978).

[17] Vishik, M. I., and Lyusternik, L. A., *The Solution of Some Perturbation Problems for Matrices and Selfadjoint or Non-selfadjoint Differential Equations I,* Russian Mathematical Surveys, Vol. 15 (1960).

HYPERSTABLE MODEL-FOLLOWING CONTROL OF AN ELECTROMAGNETIC SUSPENSION

B. C. Fabien[1]
R. W. Longman[2]
F. Freudenstien[3]
Columbia University, New York, NY 10027

ABSTRACT

This paper considers the design and simulation of an asymptotically hyperstable model-following controller for an electromagnetic suspension. A unique aspect of the controller design is that asymptotic hyperstability is achieved despite the fact that the structure of the error dynamic equations do not satisfy the Erzberger perfect model-following conditions.

I Introduction

In this paper the design of a controller for a nonlinear electromagnetic suspension is considered. The suspension system is to be one that is suitable for isolating sensitive equipment from persistent ground excitations.

It is well known that an uncontrolled electromagnetic suspension is an unstable dynamic system, in the sense that small perturbations from the equilibrium position can lead to unbounded growth in the state variables. In order for an electromagnetic suspension system to be useful it must be stable and have desirable vibration isolation characteristics.

Controllers for electromagnetic suspensions are usually designed using the linearized dynamic equations of the electromagnet [11,14,15,16]. By definition these control systems can be guaranteed to be stable only for "small" variations about the point of linearization. However, during operation the device's air-gap can vary as much as 100%, significantly affecting the values of the plant parameters. Under such conditions the stability of the control system can no longer be guaranteed.

A controller design for a partially nonlinear electromagnetic suspension was considered in [15]. However, no assessment of stability properties was presented.

The major objective of this paper is the design of an asymptotically hyperstable model-following control system for the electromagnetic suspension. The approach taken is as follows: In section II, the nonlinear dynamic equations of electromagnetic suspension are presented. In section III some major results from hyperstability theory are summarized. The design of an asymptotically hyperstable model-following controller for a class of nonlinear systems is presented in section IV. Here, it is assumed that the system dynamics are completely known and they do not satisfy the Erzberger conditions for perfect model-following. The results of section IV are used in section V to obtain an asymptotically hyperstable model-following controller for the electromagnetic suspension. Simulations of the resultant system are also presented in this section.

II Nonlinear Dynamic Equations

A schematic of the electromagnetic suspension is shown in Fig. 1. It consists of an electromagnet with its coil and a support piece, from which a load is attached. The electromagnet and the support piece are made of highly permeable magnetic materials. The current in the coil causes a force of attraction (lift-force) between

---

[1] Graduate Student. Currently at Ohio University.

[2] Professor of Mechanical Engineering.

[3] Higgins Professor of Mechanical Engineering.

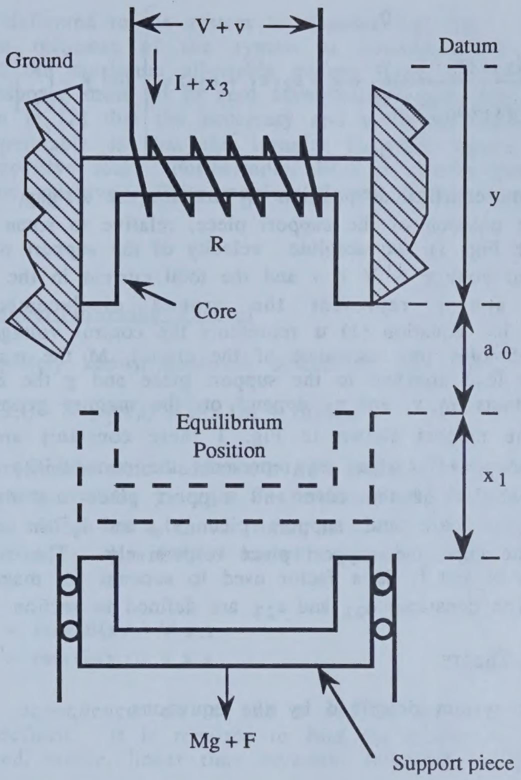


Figure 1. Schematic of the Electromagnetic Suspension

the electromagnet and the support piece. The lift-force depends inversely on the square of the air-gap and is proportional to the square of the current in the coil [8]. The lift-force is also influenced by the relative permeability and geometry of the electromagnet and the support piece, the number of turns in the coil, magnetic flux leakage and fringing effects. These factors are taken into account in the derivation of the governing equations for the system. In this paper it is also assumed that the magnetic materials are unsaturated at all times. The equations of motion can be written in state variable form as follows [8]

$$\dot{\mathbf{x}} = N_0\mathbf{x} + N_1\chi_1(\mathbf{x},t) - N_2\chi_2(\mathbf{x},t) + B(\mathbf{x},t)\mathbf{u} + H(\mathbf{x},t)\mathbf{y} \tag{1}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},\ N_0 = \begin{bmatrix} 0 & 1 & 0 \\ a_{21} & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix},\ N_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ N_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix},\ B(\mathbf{x},t) = \begin{bmatrix} 0 \\ 0 \\ \dfrac{\gamma_1+\gamma_2(x_1+a_0)}{\gamma_0} \end{bmatrix},$$

$$H(\mathbf{x},t) = \begin{bmatrix} 0 & -1 \\ 0 & 0 \\ 0 & \dfrac{-\gamma_2(x_3+I)}{\gamma_1+\gamma_2(x_1+a_0)} \end{bmatrix},\ \chi_1(\mathbf{x},t) = \begin{bmatrix} 0 \\ 0 \\ \left(\dfrac{\gamma_1+\gamma_2(x_1+a_0)}{\gamma_0}\right)\left[V + \left[\dfrac{\gamma_0\gamma_2 x_2}{(\gamma_1+\gamma_2(x_1+a_0))^2} - R\right](x_3+I)\right] \end{bmatrix},$$

$$\chi_2(x,t) = \begin{bmatrix} 0 \\ \dfrac{-\gamma_0\gamma_2(x_3+I)^2}{2M(\gamma_1+\gamma_2(x_1+a_0))^2} + \dfrac{F}{M} + g + a_{21}x_1 + a_{23}x_3 \\ 0 \end{bmatrix} \text{ and } \mathbf{y} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}.$$

In the steady state equilibrium position a_0 denotes the air-gap, V the voltage and I the current. The position of the support piece, relative to some datum, is defined as $a_0 + x_1 - y$ (see Fig. 1), the absolute velocity of the support piece is denoted by x_2, the total applied voltage is V + v and the total current in the circuit is $I + x_3$. The variables y and $\dot{y}$ represent the ground displacement and velocity respectively. Also, in equation (1) **u** represents the control voltage, i.e., **u** = v. The constant R denotes the resistance of the circuit, M the mass of the support piece, F a constant load attached to the support piece and g the acceleration due to gravity. The constants γ_0, γ_1 and γ_2 depend on the magnet geometry and material properties. For the magnet shown in Fig. 1 these constants are $\gamma_0=\mu_0 A_c N^2/2$, $\gamma_1=(l_c+l_s)/(2\mu_r)$ and $\gamma_2=1/f_1$ where μ_0 represents the permeability of free space, μ_r the relative permeability of the core and support piece material, A_c the cross-sectional area of the core and support piece, l_c and l_s the mean path of the magnetic flux in the core and support piece respectively. The number of turns in the coil is given by N and f_1 is a factor used to account for magnetic flux fringing and leakage [8]. The constants a_{21} and a_{23} are defined in section V.

III Hyperstability Theory

Consider a linear system described by the equations

$$\dot{\mathbf{e}} = A\mathbf{e} + B\mathbf{u} \tag{2a}$$

$$\mathbf{v} = C\mathbf{e} \tag{2b}$$

where the vectors **u** and **v** are of the same dimension. The pair (A,B) is completely controllable and the pair (A,C) is completely observable [9]. If the control **u**, is of the form **u** = -**w**, then hyperstability is defined as a property of the system that requires the state vector to be bounded if the inputs, **w**, are restricted to the class that satisfies the inequality

$$\int_{t_0}^{t_1} \mathbf{v}^T\mathbf{w}\, dt \geq -\delta_0^2, \ \forall\, t_1 \geq t_0 \tag{3}$$

where δ_0 is a positive constant.

The system (2) is asymptotically hyperstable if there exists a positive constant $\delta_1 > 0$, and a positive constant $\delta_0 > 0$, such that the system response $\mathbf{e}(\mathbf{e}(\mathbf{0}),t)$ satisfies

$$\|\mathbf{e}(t)\| < \delta_1\left[\|\mathbf{e}(\mathbf{0})\| + \delta_0\right], \forall\, t > 0 \tag{4a}$$

$$\lim_{t\to\infty} \mathbf{e}(t) = \mathbf{0}, \ \forall\ \mathbf{w} \text{ satisfying (3)} \tag{4b}$$

As explained in [2], the integral $\int_{t_0}^{t_1} \mathbf{v}^T\mathbf{w}\, dt$ can be interpreted as the energy delivered to the system from the outside world over the time interval $[t_0,t_1]$. Therefore, asymptotic hyperstability requires that the input control, **w**, must be chosen such that

(a) If the energy delivered to the system is bounded, i.e. (3);

(b) The maximum response of the system is bounded with respect to initial conditions and the maximum allowable energy input, i.e. (4a);

(c) The system response must go to zero asymptotically, i.e. (4b).

It is also shown in [2] that the necessary and sufficient condition for (2) to be asymptotically hyperstable is that the transfer function matrix, $H(s)=C(sI-A)^{-1}B$, must be strictly positive real. Furthermore, H(s) is strictly positive real if there exist two symmetric, positive definite matrices, P and Q such that

$$PA + A^TP = -Q \tag{5a}$$

$$B^TP = C \tag{5b}$$

IV Hyperstable Model-Following Control

Consider a completely known nonlinear system of the form

$$\dot{\mathbf{x}} = N_0\mathbf{x} + N_1\chi_1(\mathbf{x},t) - N_2\chi_2(\mathbf{x},t) + B(\mathbf{x},t)\mathbf{u} + H(\mathbf{x},t)\mathbf{y} \tag{6}$$

where $\mathbf{x}$ is the measurable state vector, $\mathbf{u} \in R^m$ denotes the control input, $\mathbf{y} \in R^p$ a measurable disturbance vector, $\chi_1, \chi_2 \in R^n$ nonlinear vectors of a known form, $B \in R^{n \times m}$ a nonlinear matrix of a known form, $H \in R^{n \times p}$ a nonlinear matrix of a known form, and $N_0, N_1, N_2 \in R^{n \times n}$ known constant matrices. Furthermore, it is assumed that

$$\mathrm{rank}(B(\mathbf{x},t)|N_1) = \mathrm{rank}(B(\mathbf{x},t)) \;\forall\; \mathbf{x},t \tag{7a}$$

$$\mathrm{rank}(B(\mathbf{x},t)|N_2) > \mathrm{rank}(B(\mathbf{x},t)) \;\forall\; \mathbf{x},t \tag{7b}$$

Henceforth the dependence on $\mathbf{x}$ and t will be omitted in the notation for variables already defined. It is required to find the control $\mathbf{u}$, such that $\mathbf{x}$ in (6) follows a prescribed, stable, linear time invariant reference model of the form

$$\dot{\mathbf{x}}_r = A_r\mathbf{x}_r + B_r\mathbf{u}_r + H_r\mathbf{y} \tag{8}$$

where the reference state vector $\mathbf{x}_r \in R^n$, the reference input $\mathbf{u}_r \in R^m$. The disturbance input $\mathbf{y}$ is the same as in (6). The matrices A_r, B_r and H_r are constant and appropriately dimensioned. Also, the pair (A_r,B_r) is completely controllable.

The error between the state vector of the reference model and the nonlinear system is defined as $\mathbf{e} = \mathbf{x}_r - \mathbf{x}$, and its derivative is $\dot{\mathbf{e}} = \dot{\mathbf{x}}_r - \dot{\mathbf{x}}$. Therefore, using (6) and (8) the error dynamic equations are given by

$$\dot{\mathbf{e}} = A_r\mathbf{e} + (A_r - N_0)\mathbf{x} + B_r\mathbf{u}_r - B\mathbf{u} + (H_r - H)\mathbf{y} - N_1\chi_1 + N_2\chi_2 \tag{9}$$

Assume that the reference input is of the form

$$\mathbf{u}_r = K_r\mathbf{x}_r \tag{10a}$$

where $K_r \in R^{n \times m}$ is constant. It is assumed that K_r is selected such that $(A_r+B_rK_r)$ is stable. The input to the nonlinear system is given by

$$\mathbf{u} = K_0(\mathbf{x},t)\mathbf{x} + K_1(\mathbf{x},t)\chi_1 + K_2(\mathbf{x},t)\chi_2 + K_3(\mathbf{x},t)\mathbf{y} \tag{10b}$$

where $K_0, K_1, K_2 \in R^{n \times m}$, and $K_3 \in R^{n \times p}$ are nonlinear.

Substituting (10) into (9) yields

$$\dot{\mathbf{e}} = (A_r + B_rK_r)\mathbf{e} + (A_r + B_rK_r - N_0 - BK_0)\mathbf{x} + (H_r - H - BK_3)\mathbf{y} - (N_1 + BK_1)\chi_1 + (N_2 - BK_2)\chi_2 \tag{11}$$

For perfect model-following $\mathbf{e} = \dot{\mathbf{e}} = \mathbf{0}$. This reduces (11) to

$$(A_r + B_rK_r - N_0 - BK_0)\mathbf{x} + (H_r - H - BK_3)\mathbf{y} - (N_1 + BK_1)\chi_1 + (N_2 - BK_2)\chi_2 = \mathbf{0} \tag{12}$$

A class of solutions to (12) are obtained by letting

$$K_0 = B^+(A_r + B_rK_r - N_0) \tag{13a}$$
$$K_1 = -B^+N_1 \tag{13b}$$
$$K_2 = B^+N_2 \tag{13c}$$
$$K_3 = B^+(H_r - H) \tag{13d}$$

where $B^+ = (B^TB)^{-1}B^T$, and A_r, B_r, K_r, H_r are selected such that

$$\text{rank}(B|(A_r + B_rK_r - N_0)) = \text{rank}(B)\ \forall\ \mathbf{x},t; \quad \text{rank}(B|(H_r - H)) = \text{rank}(B)\ \forall\ \mathbf{x},t \tag{13e}$$

Using (13) in (12), the sufficient conditions for perfect model-following control become

$$(I - BB^+)(A_r + B_rK_r - N_0) = \mathbf{0} \tag{14a}$$
$$(I - BB^+)N_1 = \mathbf{0} \tag{14b}$$
$$(I - BB^+)N_2 = \mathbf{0} \tag{14c}$$
$$(I - BB^+)(H_r - H) = \mathbf{0} \tag{14d}$$

Equations (14) are known as the Erzberger model-following conditions [5,7]. Because of assumption (7b), however, it is not possible to obtain K_2 via (13c) [13] thus, condition (14c) cannot be satisfied. For the class of nonlinear systems being considered it is assumed that the other conditions, (14a,b,d), are satisfied.

Hyperstability theory will now be applied to determine an appropriate K_2. Substitute K_0, K_1, K_3, as defined by (13), into (11) to obtain

$$\dot{\mathbf{e}} = J\mathbf{e} + I\mathbf{w}_1 \tag{15a}$$

where

$$J = (A_r + B_rK_r) \tag{15b}$$
$$\mathbf{w}_1 = -\mathbf{w} = (N_2 - BK_2)\chi_2 \tag{15c}$$

and I denotes the identity matrix.

For the error dynamics to be asymptotically hyperstable, it is required that:

(i) The linear system defined by (15) and $\mathbf{v} = P\mathbf{e}$ must be strictly positive real.

(ii) The integral inequality given by equation (3) must be satisfied.

If it is possible to satisfy these conditions, then $\lim_{t\to\infty} \mathbf{e}(t) = \mathbf{0}$. Therefore, the state vector of the nonlinear system will eventually match the state vector of the linear system. The rate at which $\mathbf{e}(t)$ approaches zero is determined by the eigenvalues of J.

To satisfy condition (i), select a P that solves the Lyapunov equation [9]

$$PJ + J^TP = -Q \tag{16}$$

where Q is a positive definite matrix. To satisfy the condition (ii), select K_2 as

$$K_2\chi_2 = \begin{cases} B^Tv(v^TBB^Tv)^{-1}(v^TN_2\chi_2 + \varepsilon_1);\ \text{if } B^Tv \neq 0 \\ 0;\ \text{if } e = 0 \text{ or } B = 0 \\ B^+(Je + \varepsilon_2);\ \text{if } B^Tv = 0 \end{cases} \tag{17}$$

where the constant scalar $\varepsilon_1 > -\delta_0^2$ and ε_2 is any constant nonzero vector $\in R^n$. To show that (17) satisfies (3) note the following conditions:

(a) If $B^Tv \neq 0$, then $v^Tw = v^TBK_2\chi_2 - v^TN_2\chi_2 = \varepsilon_1 > -\delta_0^2$ and asymptotic hyperstability is assured.

(b) If $e = 0$ then the task of model matching has been accomplished. If $B = 0$, this implies that there will be no current in the magnet coil and as a result no lift-force. Such a condition can not last for long periods of time thus eventually B will be nonzero.

(c) If v is in the left null space of B (i.e., $B^Tv = 0$) then the injection control, $B^+(Je + \varepsilon_2)$, can be used to move the system out of this space. To see this note that for the electromagnetic suspension $B^Tv \neq 0$ iff $v_3 = 0$ (see equation 1). Also, $\frac{d}{dt}(B^Tv \neq 0) \Rightarrow \dot{B}^Tv \neq 0$. $\therefore \frac{d}{dt}(B^Tv) = \dot{B}^Tv + B^TP\dot{e} = B^TP(Je + (N_2 - BK_2)\chi_2) = B^TP\varepsilon_2 \neq 0$. Thus, set $Je + (N_2 - BK_2)\chi_2 = \varepsilon_2$ to get $K_2\chi_2 = B^+(Je + \varepsilon_2)$. Here, we have used the following facts (i) $\dot{B}^Tv = 0$ since $v_3 = 0$,(ii) $B^+N_2 = 0$ and (iii) ε_2 is not in the left null space of B.

The controls used for conditions (b) and (c) above do not assure that $v^Tw > -\delta_0^2$ in general. However, conditions (b) and (c) occur only for short periods of time thus the integral inequality (3) will be satisfied over a sufficiently large time interval $[t_0,t_1]$.

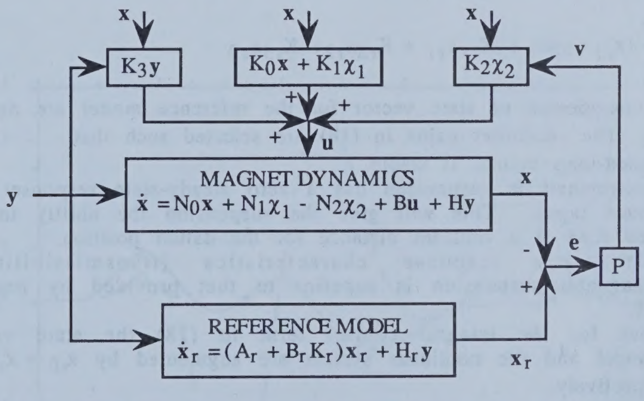


Figure 2. Structure of the Hyperstable Model-Following Controller

Note that the structure of the control law in (17) is similar to that used in [1,6,10]. However, the dynamic systems studied in [1,6,10] were required to satisfy the perfect model-following conditions. In [3,4] a control law similar to (17) was used in the control of linear systems that do not satisfy the perfect model-following conditions. However, the controller in [3,4] guarantees ultimate boundedness of the system, i.e., as t→∞ the solution to the dynamic equations will remain in some finite region, about the origin, that can be made arbitrarily small. The major assumptions and result of this section can be summarized as follows:

Assumptions:

(a) The nonlinear system (6) is completely known, and the system structure is such that (7) is satisfied.

(b) The state vector **x** and the disturbance vector **y** are measurable.

(c) The reference model is chosen such that $(A_r + B_r K_r)$ is stable and (13e), (14a,b,d) are satisfied.

Result:

(a) The asymptotically hyperstable model-following controller is given by equations (10b), (13a,b,d), (15), (16) and (17). The structure of the control system is shown in Fig. 2.

V Model-Following Controller Design and Simulation

A linear reference model of the form (8) is selected by expanding equation (1) in a Taylor series about the equilibrium position $x_1 = x_2 = x_3 = 0$, and retaining only those terms that are of first order. The resultant system of equations are

$$A_r = \begin{bmatrix} 0 & 1 & 0 \\ a_{21} & 0 & a_{23} \\ 0 & a_{32} & a_{33} \end{bmatrix}, B_r = \begin{bmatrix} 0 \\ 0 \\ b_3 \end{bmatrix} \text{ and } H_r = \begin{bmatrix} 0 & -1 \\ 0 & 0 \\ 0 & -a_{32} \end{bmatrix} \text{ where } a_{21} = \frac{\gamma_0 \gamma_2^2 I^2}{M\left(\gamma_1 + \gamma_2 a_0\right)^2},$$

$$a_{23} = \frac{-\gamma_0 \gamma_2 I}{M\left(\gamma_1 + \gamma_2 a_0\right)^2}, a_{32} = \frac{\gamma_2 I}{\left(\gamma_1 + \gamma_2 a_0\right)}, a_{33} = \frac{-R\left(\gamma_1 + \gamma_2 a_0\right)}{\gamma_0} \text{ and } b_3 = \frac{\left(\gamma_1 + \gamma_2 a_0\right)}{\gamma_0}.$$

The controller for the linear reference model has the form

$$u_r = K_{r0} \int (x_{r1} + y)dt + K_{r1}x_{r1} + K_{r2}x_{r2} + K_{r3}x_{r3} \tag{18}$$

where the components of state vector for the reference model are denoted by x_{r1}, x_{r2} and x_{r3}. The controller gains in (18) are selected such that

(a) The closed-loop system is stable.

(b) The electromagnetic suspension has a zero steady-state response to a constant disturbance input. This will give the suspension the ability to maintain the supported load at a constant distance for the datum position.

(c) The frequency response characteristics (transmissibility) of the electromagnetic suspension is superior to that provided by passive isolation systems.

To account for the integral control term in (18) the state vectors of the reference model and the nonlinear system are augmented by $\dot{x}_{r0} = x_{r1} + y$ and $\dot{x}_0 = x_1 + y$ respectively.

Now, consider an electromagnetic suspension with the following design parameters [8], $\gamma_0 = 1.27 \times 10^{-3}$, $\gamma_1 = 2.623 \times 10^{-5}$, $\gamma_2 = 8.333 \times 10^{-1}$, $M = 1.649$, $R = 5$, $F = 445$, $g = 9.8$, $I = 2$ and $V = 10$.

The controller gains for the linear reference model are selected as: $K_{r0} = 2.484 \times 10^2$, $K_{r1} = 2.231 \times 10^4$, $K_{r2} = 2.478$, $K_{r3} = -2.364 \times 10^1$.

To complete the design of the model-following controller, the solution to the Lyapunov equation (16) must be found. Using $Q = \text{diag}(10,10,10,10)$ and solving (16) gives the following result:

$$P = \begin{bmatrix} 3.66 \times 10^{7} & 2.15 \times 10^{6} & 4.44 \times 10^{4} & -1.19 \times 10^{-2} \\ & 1.80 \times 10^{10} & 4.00 \times 10^{6} & -2.31 \times 10^{7} \\ \text{SYMMETRIC} & & 1.02 \times 10^{3} & -5.12 \times 10^{3} \\ & & & 2.96 \times 10^{4} \end{bmatrix}$$

The nonlinear electromagnet suspension and its model-following controller are simulated to evaluate the performance of the system. The response of the system to a disturbance input is examined. The input disturbance and initial conditions are as follows:

$$\left.\begin{array}{l} y(t) = 0.001 \\ \dot{y}(t) = 0 \end{array}\right\} 0 < t < \infty,\ \mathbf{x}(0) = \mathbf{0}.$$

The response of the nonlinear model with hyperstable model-following control (M1) is compared to the response of the linear reference model with linear feedback control (M2), and the response of the nonlinear model with linear feedback control (M3). These comparisons indicate how well the hyperstable model-following controller is able to track the linear model, and the performance degradation incurred by using linear feedback control on the nonlinear system.

The simulation results are illustrated in Fig. 3. In these plots $Ei = x_{ri} - x_i$, represents the error in the $(i)^{\text{th}}$ state variable between M2 and M1, and Eni the error in the $(i)^{\text{th}}$ state variable between M2 and M3. Note that the dynamic equations are integrated using a fourth order Runge-Kutta algorithm with automatic step size correction [12].

The results shown in Fig. 3 indicate that the model-following controller is able to track the linear reference model very closely. Whereas, linear feedback control of the nonlinear model is not as effective. It can be seen that the peak-to-peak error between M2 and M3 is more than twice the peak-to-peak error between M1 and M2.

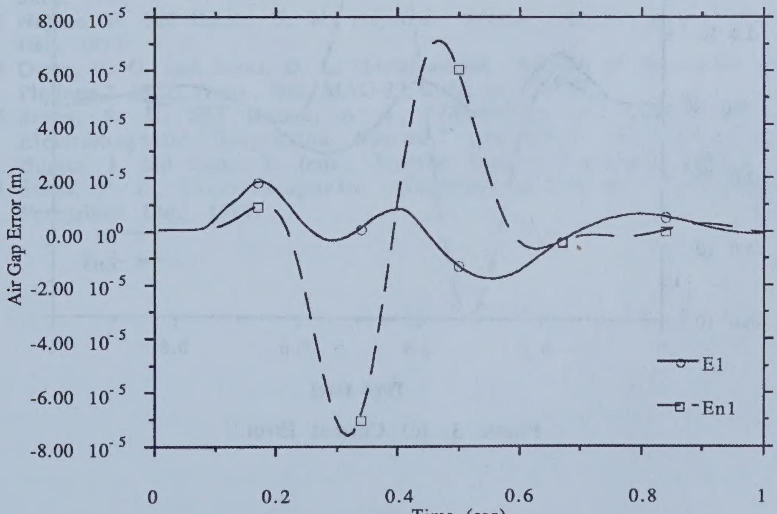


Figure 3. (a) Air Gap Error.

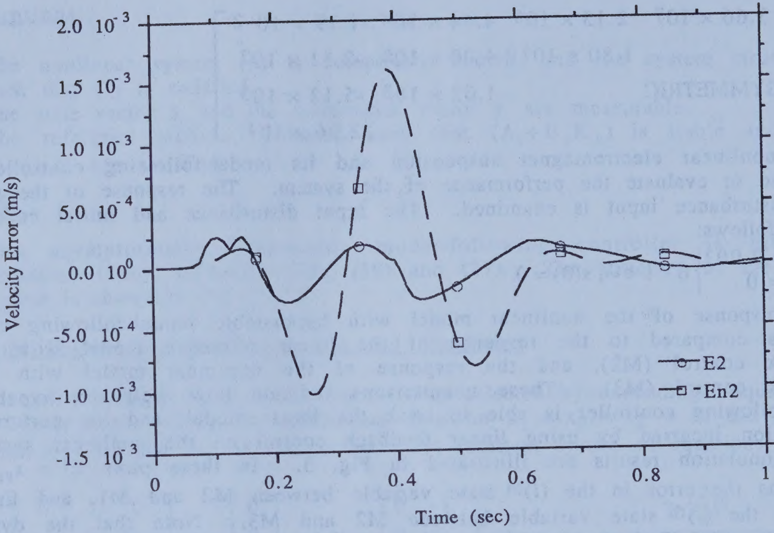


Figure 3. (b) Velocity Error.

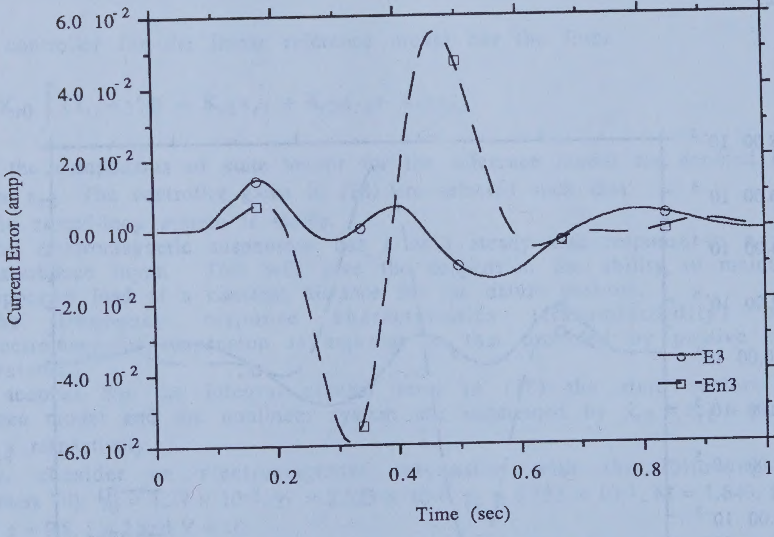


Figure 3. (c) Current Error.

VI ACKNOWLEDGEMENT

The support of this research by General Motors Research Laboratories through a research contract with Columbia University is hereby acknowledged.

VII REFERENCES

1 Ambrosino, G., Celentano, G. and Garofalo, F., "Adaptive Tracking Control of Industrial Robots," *ASME Trans., J. Dyn. Sys., Meas., and Cntl.*, Vol. 110, 1988, pp. 215-220.

2 Anderson, B. D. O., "A Simplified Viewpoint of Hyperstability," *IEEE Trans. Automatic Control*, Vol. AC-22, pp. 83-88, 1977.

3 Chen, Y. and Leitmann, G., "Robustness of uncertain systems in the absence of matching assumptions," *Intl. J. Control*, Vol. 45, pp. 1527-1542.

4 Chen, Y., "On the Robustness of Mismatched Uncertain Dynamical Systems," *ASME Trans., J. Dyn. Sys., Meas., and Cntl.*, Vol. 109, pp. 29-35.

5 Chen, Y. T., "Perfect Model Following with a Real Model," *JACC*, 1973, pp. 287-293.

6 Corless, M., "Tracking Controllers for Uncertain Systems: Application to a Manutec R3 Robot," *ASME Trans., J. Dyn. Sys., Meas., and Cntl.*, Vol. 111, 1989, pp. 609-618.

7 Erzberger, H., "Analysis and Design of Model Following Systems by State Space Techniques," *JACC*, 1968, pp. 572-581.

8 Fabien, B. C., "Contributions to the Design Optimization, Control and Synthesis of Electromechanical Systems", Ph.D. Dissertation, Columbia University, 1990.

9 Friedland, B., Control System Design: An Introduction to State Space Methods, McGraw-Hill, 1986.

10 Gutman, S., "Uncertain Dynamical Systems - A Lyapunov Min-Max Approach," *IEEE Trans.*, Vol. AC-24, 1979, pp. 437-443.

11 Jones, D. I. and Owen, R. G., "A Magnetically Levitated Anti-Vibration Mount," *IEEE Trans.*, Vol. MAG-20, 1984, pp. 1687-1689.

12 Kreyszig, E., *Advanced Engineering Mathematics*, 5th Edition, John Wiley & Sons, 1983.

13 Nobele, B. and Daniel, J. W., *Applied Linear Algebra*, 2nd Edition, Prentice-Hall, 1977.

14 Owen, R. G. and Jones, D. I., "Multivariable Control of An Active Anti-Vibration Platform," *IEEE Trans.*, Vol. MAG-22, 1986, pp. 523-525.

15 Sinha, P. K. and Hulme, A. J., "Simulation and Control of a Nonlinear Electromagnetic Suspension System," *Simulation in Engineering Sciences*, Burger, J. and Jarny, Y. (eds.), Elsevier Science Publishers, 1983.

16 Sinha, P. K., *Electromagnetic Suspension: Dynamics and Control*, Peter Peregrinus Ltd., 1987.

MODELING AND CONTROL OF RAPID DEPLOYMENT TRUSS STRUCTURE SYSTEMS [1]

M. BOESCH, J.X. XU and ÜMIT ÖZGÜNER
Department of Electrical Engineering
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210

ABSTRACT

This paper introduces a Rapid Deployment Truss (RDT) structure, a flexible structure which is to be deployed from a stowed configuration at high speed. Because RDT holonomic constraints maintain structure connectivity, a method is developed for modeling flexible closed chain linkages, by extending a method known for rigid links. Numerical simulations suggest validity of the method in satisfying the constraint equations. Different control approaches are considered, including a simple PD control and a frequency shaped variable structure control.

I. INTRODUCTION

This paper considers the modeling and control of a truss structure deployed from a stowed position at high speed. The structure is primarily spring deployed, but its speed and associated vibrations are to be regulated with joint actuators. This is the *Rapid Deployment Truss (RDT) problem*, and it is necessary to incorporate such elements in large space structures as part of parabolic dish antenna back-up structures, instrument support platforms, and other space based configurations.

Envisioned structures are characterized by densely packed, low frequency, lightly damped modes, and are subject to nonlinear couplings. The planar RDT of Figure 1 shows many of the generic features of such space structure problems. Previous work on

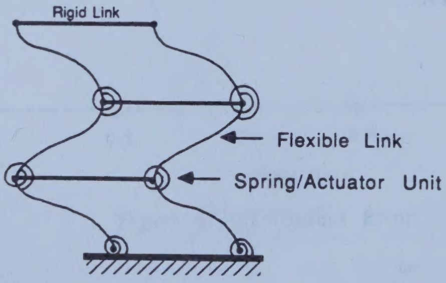


Figure 1: Basic Rapid Deployment Truss

---

[1] Supported by the Lawrence Livermore National Laboratories, under Subcontract B083599.

deployable structures emphasizes kinematics of such structures. A flat-folding octahedral truss is experimentally verified in [1], but the system dynamics are dominated by dynamics of lead-screw type actuators. Kinematics of a planar variable geometry truss are formulated in [2], but not dynamics. In these cases, structural vibrations during the deployment phase are not addressed, and in fact the deployment is deliberately slow in an attempt to not excite structural modes excessively.

Motion of flexible structures undergoing large angle, fast slewing maneuvers has been investigated in the context of flexible manipulators. Research on flexible planar manipulators, while fairly mature, has been limited to devices with one or two flexible links, with each link usually modeled as a single flexible beam.

In modeling flexible links as slender beams, it is both convenient and a valid approximation to restrict the model to finite order by truncating the full set of modes and including only those modes which are considered most important based upon some criterion. For simple slewing structures undergoing small vibrations the continuous solution to a partial differential equation (PDE) representative of its motion can be found as in [3] for both one- and two-link flexible open kinematic chains (manipulators). However, even when the exact PDE solution is available, it generally must be spatially discretized in some manner for purpose of simulation. In many cases it can be shown by limiting criteria such as controller bandwidth that it is not feasable to attempt to model or control a very large number of modes.

II. MODELING OF RDT

The basic truss structure studied is that of a planar 2-bay box type truss similar to Figure 1. Note in this figure that torque springs and control actuators are chosen to act between serially connected flexible longitudinal members, while lateral members or "rungs" are end connected as pinned, and thus no moments can be transferred from the longitudinal members to these. The resulting assumed structure becomes similar to two two-link flexible manipulators operating in parallel, with deployment springs and holonomic constraints imposed by rungs at joints. Control will be complicated due to the constraint forces acting to couple the sides. Inclusion and control of constraint forces for rigid link systems is developed by Hemami in [4], and in this paper is extended to flexible bodies so as to build up the RDT model.

A. General Solution of Holonomic Constraint Forces

In this section a method proposed by Hemami in [5] is derived which keeps holonomic (configuration dependent) constraint forces in evidence for modeling of closed-chain linkages. The derived method is applicable to rigid or flexible linkages, if the flexure variables have been expressed as a finite dimensional summation of modal terms, here based upon the assumed modes modeling method.

If vector z is taken as the position state vector of the complete unconstrained system, then the corresponding linkage dynamics can always be expressed as

$$M\ddot{z} + g(z, \dot{z}) = \left(\frac{\partial C}{\partial z}\right)^T \Gamma + u \tag{1}$$

where Γ is the vector of constraint forces, and the holonomic constraints are expressed as $C(z) = 0$. To solve (1) explicitly for constraint forces Γ, first twice differentiate C with respect to time, giving

$$\ddot{C}(z) = \frac{d}{dt}\left(\frac{\partial C}{\partial z}\right)\dot{z} + \frac{\partial C}{\partial z}\ddot{z} = 0. \tag{2}$$

Solve for acceleration $\ddot{z}$ in (1) to find

$$\ddot{z} \;=\; M^{-1}\left[-g(z,\dot{z}) + \left(\frac{\partial C}{\partial z}\right)^T \Gamma + u\right]. \tag{3}$$

Multiply through (3) by $(\frac{\partial C}{\partial z})$ then substitute from (2) for $(\frac{\partial C}{\partial z})\ddot{z}$ which gives

$$-\frac{d}{dt}\left(\frac{\partial C}{\partial z}\right)\dot{z} = -\frac{\partial C}{\partial z}M^{-1}g(z,\dot{z}) + \frac{\partial C}{\partial z}M^{-1}\left(\frac{\partial C}{\partial z}\right)^T \Gamma + \frac{\partial C}{\partial z}M^{-1}u. \tag{4}$$

Solving for Γ yields

$$\Gamma = \left[\frac{\partial C}{\partial z}M^{-1}\left(\frac{\partial C}{\partial z}\right)^T\right]^{-1}\left\{-\dot{z}^T\frac{\partial}{\partial z}\left[\left(\frac{\partial C}{\partial z}\right)\dot{z}\right] + \frac{\partial C}{\partial z}M^{-1}\left[g(z,\dot{z}) - u\right]\right\}, \tag{5}$$

as the expression of constraint forces.

B. *Dynamical Consideration of a Simplified RDT Structure*

Consider modeling of a simplified RDT structure, dynamically equivalent to parallel rigid-flex manipulators constrained in position by a rung at the endpoints of the first links only. A detailed structural schematic appears in Figure 2. Parameters appearing in the figure are defined in Table 1. For each rigid-flex manipulator individually, equations of motion using the assumed modes constrained expansion will result in

$$M\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \ddot{q}_2\end{bmatrix} + F_r\begin{bmatrix}\dot{\theta}_1^2\\ \dot{\theta}_2^2\\ \dot{\theta}_1\dot{\theta}_2\end{bmatrix} + F_{rf}\begin{bmatrix}\dot{\theta}_1\dot{q}_2\\ \dot{\theta}_2\dot{q}_2\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \mu_2\dot{q}_2\end{bmatrix} + K\begin{bmatrix}\theta_1\\ \theta_2\\ q_2\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}, \tag{6}$$

where modal states $q_2 = [q_{21}, q_{22}, \ldots, q_{2j}, \ldots, q_{2Ni}]^T$, j = mode number $(1, \ldots, N_2)$, and M, F_r, F_{rf}, and K are mass, centrifugal, coriolis, and stiffness matrices. A small damping term involving coefficients μ_2 is included for modal states. Control torques u_1 have no direct effect on modal state variables, but affect the rigid body angles θ_1, and through coupling off-diagonal terms in the mass matrix affect modal states.

The rung equations of motion can be written using balance of inertia and constraint forces in the free body diagram of Figure 3. Summing horizontal (X_0 direction) and vertical (Y_0 direction) forces gives

$$
\begin{aligned}
f_1 &= m^* \ddot{x}_1^* - f_1^{'}, \\
g_1 &= m^* \ddot{y}_1^* - g_1^{'},
\end{aligned}
\qquad (7)
$$

where $\ddot{x}_1^*$ and $\ddot{y}_1^*$ are the horizontal and vertical acceleration components of the center of mass of rung. Likewise summing moments about the center of mass (θ_1^* direction) gives

$$f_1 d_1^* s_1^* - f_1^{'}(d - d_1^*)s_1^* + g_1^{'}(d - d_1^*)c_1^* - g_1 d_1^* c_1^* = I_1^* \ddot{\theta}_1^*, \qquad (8)$$

where $s_1^{[\cdot]}$ and $c_1^{[\cdot]}$ represent $\sin \theta_1^{[\cdot]}$ and $\cos \theta_1^{[\cdot]}$ respectively.

Angular state variable θ_1^* will be maintained explicitly in final equations of motion, as the degree of freedom which allows calculation of explicit constraint forces $\{f_1^{'}, g_1^{'}\}$, but accelerations $\ddot{x}_1^*$ and $\ddot{y}_1^*$ can be expressed in terms of the left hand side manipulator's state variables and θ_1^*, by writing equations for the center of mass position of the rung, then twice differentiating these with respect to time, giving accelerations

$$
\begin{aligned}
\ddot{x}_1^* &= -L_1 \dot{\theta}_1^2 c_1 - d_1^* \dot{\theta}^{*2} c_1^* - L_1 \ddot{\theta}_1 s_1 - d_1^* \ddot{\theta}^* s_1^*, \\
\ddot{y}_1^* &= -L_1 \dot{\theta}_1^2 s_1 - d_1^* \dot{\theta}^{*2} s_1^* + L_1 \ddot{\theta}_1 c_1 + d_1^* \ddot{\theta}^* c_1^*.
\end{aligned}
\qquad (9)
$$

Substitute the preceding accelerations into force balance equations (7) to yield

$$
\begin{aligned}
f_1 &= m^*(-L_1 \dot{\theta}_1^2 c_1 - d_1^* \dot{\theta}^{*2} c_1^* - L_1 \ddot{\theta}_1 s_1 - d_1^* \ddot{\theta}^* s_1^*) - f_1^{'} \\
g_1 &= m^*(-L_1 \dot{\theta}_1^2 s_1 - d_1^* \dot{\theta}^{*2} s_1^* + L_1 \ddot{\theta}_1 c_1 + d_1^* \ddot{\theta}^* c_1^*) - g_1^{'}.
\end{aligned}
\qquad (10)
$$

To write the equations of motion of the rung in terms of left manipulator variables and the explicit constraint forces $\{f_1^{'}, g_1^{'}\}$, substitute the right half sides of (10) into (8) to have

$$\ddot{\theta}^*(I^* + d^{*2} m^*) = m^* d_1^* [\ddot{\theta}_1(-L_1 c(\theta_1 - \theta_1^*)) + \dot{\theta}_1^2 L_1 s(\theta_1 - \theta_1^*)] - d s_1^* f_1^{'} + d c_1^* g_1^{'}. \qquad (11)$$

The implicit $\{f_1, g_1\}$ and explicit $\{f_1^{'}, g_1^{'}\}$ constraint forces will affect the individual manipulator dynamics through a term added to each manipulator's equation of motion (6) to give equations for the left and right side manipulators respectively of the forms

$$M[\cdot] + F_r[\cdot] + F_{rf}[\cdot] + \text{Damp} + K[\cdot] = [\cdot]u - J^T[f_1,\ g_1]^T \qquad (12)$$

$$M^{'}[\cdot] + F_r^{'}[\cdot] + F_{rf}^{'}[\cdot] + \text{Damp}^{'} + K^{'}[\cdot] = [\cdot]u^{'} - J^{'T}[f_1^{'},\ g_1^{'}]^T \qquad (13)$$

where the first links' Jacobian matrices J and $J^{'}$ expressed in terms of inertial coordinates $\{X_0, Y_0\}$ have the forms

$$J^T = \begin{bmatrix} -s_1 L_1 & c_1 L_1 \\ 0 & 0 \\ 0_{N_2} & 0_{N_2} \end{bmatrix}, \qquad J^{'T} = \begin{bmatrix} -s_1^{'} L_1^{'} & c_1^{'} L_1^{'} \\ 0 & 0 \\ 0_{N_2^{'}} & 0_{N_2^{'}} \end{bmatrix}. \qquad (14)$$

where 0_{N_2} signifies a null vector of dimension N_2. In the equation of dynamics for the left side manipulator (12) the expressions of forces f_1 and g_1 from (10) can be utilized so that these forces are included *implicitly* in the left manipulator model. Here the term $-J^T[f_1,\ g_1]^T$ will have only first row nonzero elements of the following form.

$$s_1 L_1 f_1 - c_1 L_1 g_1 = m^*[d_1^* \dot{\theta}^{*2} L_1 s(\theta_1^* - \theta_1) - \ddot{\theta}_1 L_1^2 - \ddot{\theta}^* d_1^* L_1 c(\theta_1^* - \theta_1)] - s_1 L_1 f_1^{'} + c_1 L_1 g_1^{'}. \qquad (15)$$

The equation (15) is easily included in the left side manipulator equations (12) by suitably modifying elements of the mass, centrifugal, and coriolis matrices therein. The equation of (12) with only explicit constraint forces $\{f_1^{'}, g_1^{'}\}$ then takes on the form

$$M[\cdot] + F_r[\cdot] + F_{rf}[\cdot] + \text{Damp} + K[\cdot] = [\cdot]u + J^T[f_1^{'},\ g_1^{'}]^T \tag{16}$$

For inclusion of the rung into the system model, equation (11) is appended as a row to the dynamics of the left manipulator (16).

In this simplified RDT structure since the constraint loop is made up of only rigid members, the Jacobians are not dependent upon modal terms as would be the case for a flexible link loop. The method is however extendable to flexible members in the constraint loop as shown in [6]. If the two bay RDT first links were also flexible, following the same procedure of this section gives rung equation of motion

$$\begin{aligned} \ddot{\theta}^*(I^* + d^{*2}m^*) = m^*d_1^*\{&\ddot{\theta}_1[\alpha_1 s(\theta_1 - \theta_1^*) - L_1 c(\theta_1 - \theta_1^*)] + \ddot{\alpha}_1[-c(\theta_1 - \theta_1^*)] + \\ &+\dot{\theta}_1^2[L_1 s(\theta_1 - \theta_1^*) + \alpha_1 c(\theta_1 - \theta_1^*)] + \dot{\alpha}_1\dot{\theta}_1 2s(\theta_1 - \theta_1^*)\} + \\ &- ds_1^* f_1^{'} + dc_1^* g_1^{'}, \end{aligned} \tag{17}$$

and Jacobians

$$J^{[']T} = \begin{bmatrix} -s_1^{[']}L_1^{[']} - c_1^{[']}\sum_{i=1}^{N_1^{[']}} \phi_{1i}^{[']}(x)q_{1i}^{[']}(t) & c_1^{[']}L_1^{[']} - s_1^{[']}\sum_{i=1}^{N_1^{[']}} \phi_{1i}^{[']}(x)q_{1i}^{[']}(t) \\ 0 & 0 \\ -s_1^{[']}\phi_{11}^{[']}(x) & c_1^{[']}\phi_{11}^{[']}(x) \\ \vdots & \vdots \\ -s_1^{[']}\phi_{1N_1}^{[']}(x) & c_1^{[']}\phi_{1N_1}^{[']}(x) \\ 0_{N_2^{[']}} & 0_{N_2^{[']}} \end{bmatrix}, \tag{18}$$

in which endpoint flexure $\alpha_i^{[']}$ is expressed as a product of modal coordinates and mode shapes by appealing to the the assumed modes method, and in which the influence of flexure terms is evident.

Prior to solving (5) for constraint forces, the method discussed in Section A. requires writing constraint equations of the form $C(z) = 0$ where z is the position state vector of the complete unconstrained system. If endpoints of left and right side manipulators' first links are expressed in inertial coordinates as $\bar{x}_1$ and $\bar{x}_1^{'}$ where

$$\bar{x}_1 = \begin{bmatrix} L_1 c_1 \\ L_1 s_1 \end{bmatrix}, \qquad \bar{x}_1^{'} = \begin{bmatrix} d + L_1^{'}c_1^{'} \\ L_1^{'}s_1^{'} \end{bmatrix}, \tag{19}$$

then the constraint equation can be written as

$$C(z) = \bar{x}_1^{'} - \bar{x}_1 - d[c_1^*,\ s_1^*]^T = 0. \tag{20}$$

The model of this simplified RDT structural dynamics is thus shown to include explicitly holonomic constraint forces as a means for coupling its sides.

C. *State Space Formulation*

The equations of motion for rigid and flexible RDT type structures, can be written in the form

$$M(z)\ddot{z} + g(z, \dot{z}) = \left(\frac{\partial C}{\partial z}\right)^T \Gamma(z, u) + \bar{B}u \tag{21}$$

where z is the $n/2$ by 1 vector of rigid-body and flexure coordinates, $M(z)$ is the positive definite mass matrix, $g(z,\dot{z})$ is all nonlinear terms due to centrifugal, coriolis, and gravity effects, the holonomic constraint expression $C(z) = 0$ represents all closed structural loops, $\Gamma(z,u)$ is a vector of constraint forces or Lagrange multipliers, and $\bar{B}$ is the control distribution matrix for input vector u. Using the expression derived for constraint forces $\Gamma(z,u)$ in (5), grouping terms in input u, letting $x = [z, \dot{z}]^T$, with $x \in R^n$ and the system dynamical equation (21) can be rewritten as

$$\dot{x} = f(x) + B(x)u \tag{22}$$

where $f(x) =$

$$\begin{bmatrix} \dot{z} \\ M^{-1}(z)\left(-g(z,\dot{z}) - \left(\frac{\partial C}{\partial z}\right)^T \left[\frac{\partial C}{\partial z} M^{-1} \left(\frac{\partial C}{\partial z}\right)^T\right]^{-1} \left\{\frac{d}{dt}\left(\frac{\partial C}{\partial z}\right)\dot{z} - \frac{\partial C}{\partial z} M^{-1} g(z,\dot{z})\right\}\right) \end{bmatrix}, \tag{23}$$

$$B(x) = \begin{bmatrix} 0 \\ M^{-1}(z)\left\{\left(\frac{\partial C}{\partial z}\right)^T \left[\frac{\partial C}{\partial z} M^{-1} \left(\frac{\partial C}{\partial z}\right)^T\right]^{-1} \left[-\frac{\partial C}{\partial z} M^{-1}\right] + \bar{B}\right\} \end{bmatrix}. \tag{24}$$

Since matrix M is a function of x, both f and B are nonlinear and time varying.

III. NUMERICAL RDT SIMULATIONS

The equations simulated are those of identical two link open-kinematic chains as given in [3] but spatially discretized using the assumed modes method, and suitably modified to include rungs as given in Section B. The simulated system parameters are defined in Table 1, and numerical values appear in Table 2.

All simulations are of a stowed RDT type apparatus as in Figure 4 suddenly deploying and attempting to reach its final deployed position as in Figure 5. The numerical values of initial and final conditions of the motion are listed in Table 3, page 8. These chosen conditions are somewhat contrived, so as to not generate Jacobian matrices of less than full rank. This loss of rank would cause the matrix $\frac{\partial C}{\partial z} M^{-1} \left(\frac{\partial C}{\partial z}\right)^T$ in (5) to become singular, making a solution for Γ, the constraint forces, impossible. Simulations were carried out maintaining the smallest nonzero number possible on the order of $\epsilon = 10^{-18}$ as a measure of numerical accuracy. Results follow for the rigid-flex RDT and the flex-flex RDT simulations.

A. Open Loop Simulations

Open loop simulation of the simplified rigid-flex RDT appear in Figures 6 through 9, in which only time responses of left side of the RDT will be shown. For these simulations $E_1^{[']} I_1^{[']} = \infty$ and $N_1^{[']} = 0$. On all time responses with two y-axis subscripts the first subscript names the solid line plot and the second the dotted line.

Figure 6 of rigid body motion clearly shows the absence of damping in these dynamical terms. Time response of θ_1^* (not included) is on the order of ϵ. Figure 7 shows time responses for position of flexure coordinate α_2 of the end RDT links. Figure 8 displays horizontal constraint force, and Figure 9 gives error residual in horizontal constraint equation $C(z) = 0$, here of order ϵ.

Open loop simulation of a flex-flex RDT appear in Figures 10 through 14, in which only time responses of left side of the RDT will be shown. For these simulations $E_1^{[']} I_1^{[']} = 8.8778$ and $N_1^{[']} = 1$. Figure 10 of rigid body motion again shows absence of damping in these terms. Time response of θ_1^* (not included) is on the order of 10^{-4} . Figure 11 shows time responses for position of flexure coordinate α_1 of the first RDT links, while Figure 12 shows the same for α_2 of the end RDT links. Figure 13 displays horizontal constraint force, and Figure 14 gives error residual in horizontal constraint equation $C(z) = 0$, here of order 10^{-12}. Although the error residual is larger than ϵ, it is still several orders of magnitude less than either rigid body or flexure coordinate amplitudes.

B. Closed Loop Simulations

The flex-flex RDT is simulated with a simple individual joint PD control in which

$$u_i = \gamma_{pi}(\theta_i - \theta_{di}) + \gamma_{vi}\dot{\theta}_i \tag{25}$$

where θ_{di} is the desired joint final position. In this section the deployment springs are used to supply all deployment energy so the proportional gain term $\gamma_{pi} = 0$ and derivative gains are set to $\{-10.0, \ -.02\}$ for the first and second links respectively.

Figure 15 shows the rigid body motion, indicating that finite settling time is achieved. Figure 16 shows time responses for position of flexure coordinate α_1 of the first RDT links, while Figure 17 shows the same for α_2 of the end RDT links. Figure 18 shows the control efforts of the first and the second links respectively.

IV. CONCLUSION

This paper describes a systematic approach to achieve a realistic model of a Rapid Deployment Truss structure by regarding it as two multi-linkage flexible manipulators which are coupled to each other through a number of pinned-pinned rigid lateral links. Coupling effects are modeled using a method derived for calculation of holonomic constraint forces. For slewing flexible structures where the vibrational modes are not to be excited, the modeling approach provides a method for simulating constrained flexible structures.

Simulation results are presented of simple RDT examples. The simulated method maintains holonomic constraints to the order of calculational precision if the constraint loop is connected rigid members, while a constraint loop of connected flexible members has a small but somewhat larger residual. As for future research on RDT, a general planar RDT model is being simulated including several modes in each link, and several holonomic constraints, and under the control of a frequency shaped variable structure controller [7].

V. ACKNOWLEDGEMENT

Discussions with Professor H. Hemami are gratefully acknowledged.

VI. REFERENCES

[1] R. Wynn, H. Robertshaw, and C. Horner, “An analytical study of a six degree-of-freedom active truss of use in vibration control,” in *Proc. AIAA/ASME/ASCE/AHS/ASC 31st Structures, Structural Dynamics, and Materials Conf.*, Long Beach, CA, Apr. 1990.

[2] S. Matunaga, K. Miura, and M. Natori, “A construction concept of large space structures using intelligent/adaptive structures,” in *Proc. AIAA/ASME/ASCE/AHS/ASC 31st Structures, Structural Dynamics, and Materials Conf.*, Long Beach, CA, Apr. 1990.

[3] F. Khorrami, “Asymptotic perturbation and lyaponov stability based approaches for control of flexible and rigid robot manipulators,” Ph.D. Thesis, The Ohio State University, 1988.

[4] H. Hemami, “A state space model for interconnected rigid bodies,” *IEEE Trans. on Automatic Control*, vol. AC-27, no. 2, , Apr. 1982.

[5] H. Hemami and B. F. Wyman, “Modeling and control of constrained dynamic systems with application to biped locomotion in the frontal plane,” *IEEE Trans. on Automatic Control*, vol. AC-24, no. 4, , Aug. 1979.

[6] M. A. Boesch, “Design and modeling aspects of a rapid deployment truss structure,” M.S. Thesis, The Ohio State University, 1990.

[7] K. D. Young and Ü. Özgüner, “Frequency shaped variable structure control,” in *Proc. of the ACC*, pp. 181–185, San Diego, CA, May 1990.

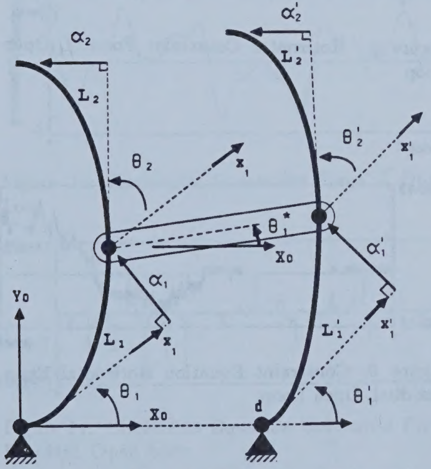


Figure 2: Simplified RDT Structural Schematic

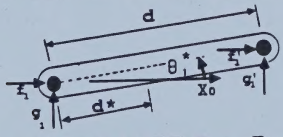


Figure 3: Simplified RDT Rung Free-body Diagram

Table 1: Simulation Parameter Definitions

I_h	hub inertia
$L_i^{[']}$	length of link $i^{[']}$
$M_i^{[']}$	mass of link $i^{[']}$
d_i	distance between first links of RDT, length of rung
m^*	mass of rung
d_i^*	distance from rung left joint to center of mass
$l_i^{[']}$	spatial variable for link $i^{[']}$
$\alpha_i^{[']}(l_i, t)$	angle of flexure of link $i^{[']}$ at location l_i
$\theta_i^{[']}$	rigid link angle of link $i^{[']}$
$\rho_i^{[']}$	mass density of link $i^{[']}$
$E_i^{[']} I_i^{[']}$	stiffness term for link $i^{[']}$
$\hat{u}_i^{[']}$	input torque at joint $i^{[']}$
$N_i^{[']}$	number of modes of link $i^{[']}$
$M_{hub}^{[']}$	hub mass
$k_{\tau i}^{[']}$	torsional spring constant at joint $i^{[']}$
$\mu_{ij}^{[']}$	damping of mode j of link $i^{[']}$
$\gamma_{pi}^{[']}$	proportional gain at joint $i^{[']}$
$\gamma_{vi}^{[']}$	derivative gain at joint $i^{[']}$

Table 2: Simulation Parameters

i	$1,1'$	$2,2'$
$I_{hi}\,[kg \cdot m^2]$	1.5	0.0
$M_{hi}\,[kg]$	2.8	0.0
$L_i\,[m]$	0.75	0.5
$E_i I_i\,[N \cdot m^2]$	8.877	0.2183
$M_i\,[kg]$	1.2	0.055
N_i	1	2
$k_{\tau i}\,[N/rad]$	10.0	0.1
$\mu_i\,[N \cdot s^2/m]$	0.973	0.973
$d\,[m]$	0.75	0
$d^*\,[m]$	0.375	0
$m^*\,[kg]$	1.2	0

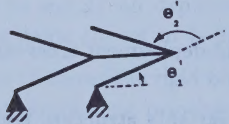


Figure 4: Stowed Initial Condition of Simulation

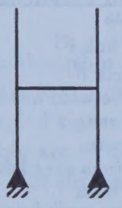


Figure 5: Deployed Final Condition of Simulation

Table 3: Initial and Final Conditions

	θ_1 deg	θ_1' deg	θ_2 deg	θ_2' deg	α_1 m	α_2 m
i.c.	5.0	5.0	175.0	175.0	0	0
f.c.	90.0	90.0	0	0	0	0

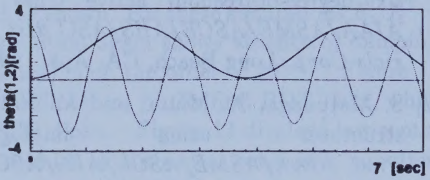


Figure 6: Rigid Body Angular Positions, Open Loop

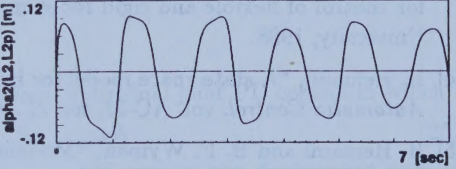


Figure 7: End Links Tip Flexure, Open Loop

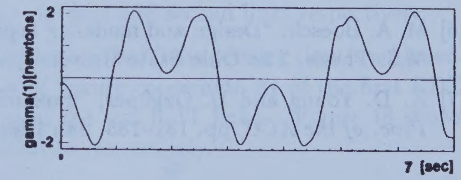


Figure 8: Horizontal Constraint Force f_1',Open Loop

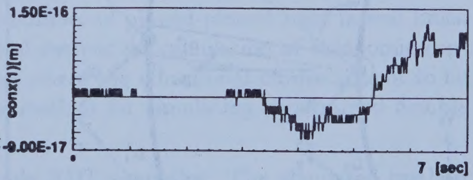


Figure 9: Constraint Equation Horizontal Error Residual, Open Loop

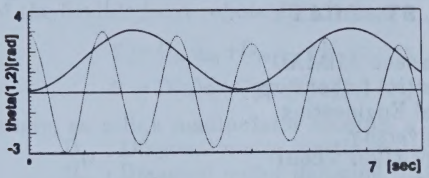


Figure 10: Rigid Body Angular Positions, Open Loop

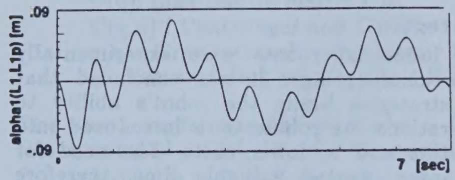


Figure 11: First Links Tip Flexure, Open Loop

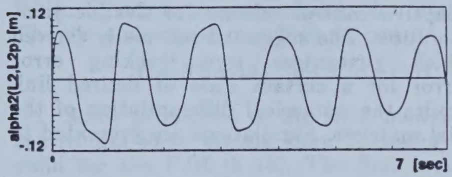


Figure 12: End Links Tip Flexure, Open Loop

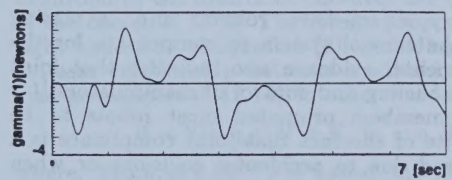


Figure 13: Horizontal Constraint Force f_1',Open Loop

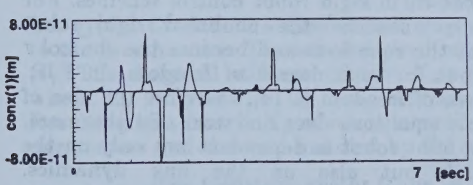


Figure 14: Constraint Equation Horizontal Error Residual, Open Loop

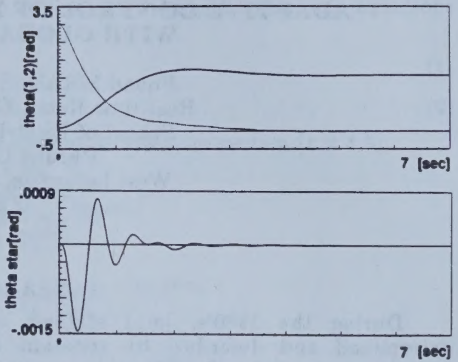


Figure 15: Rigid Body Angular Positions, Closed Loop

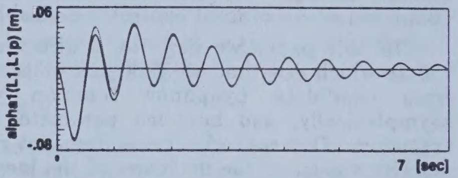


Figure 16: First Links Tip Flexure, Closed Loop

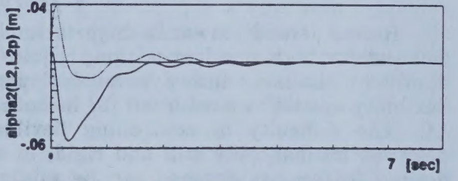


Figure 17: End Links Tip Flexure, Closed Loop

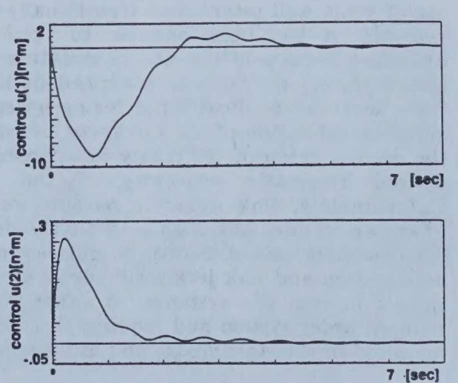


Figure 18: Control efforts, Closed Loop

ADAPTIVE CONTROL OF FLEXIBLE JOINT ROBOTS WITH GLOBAL STABILITY

Fouad MRAD, Shaheen AHMAD
Real-time Robot Control Laboratory
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907 - 0501
U S A

1. Abstract

During the 1980's, joint stiffness of industrial robots were experimentally determined and described by constant torsional springs. It was concluded that neglecting the joint flexibility in control strategies limits the robot's ability to perform high speed and high precision operations. As robots were introduced into industrial environments to increase production and to lower costs. The need for reprogramming with different loads and tasks, wasted valuable time, therefore adaptive instead of fixed control is desirable.

In this paper we describe a direct adaptive control scheme for flexible joint robots which does not use link jerk or acceleration. The suggested scheme is derived from candidate Lyapunov function which guarantees zero tracking error asymptotically, and bounded parameter error for a certain class of desired link trajectory. Our control scheme does not require the numerical differentiation of the velocity signal, or the inversion of the inertial matrices. Simulations are presented to verify the validity of the control scheme.

2. Introduction

Recent trend towards high-technology direct-drive robots, and increasing demand for high-speed operation, require the control system to compensate for the dominant nonlinear link dynamics. Experimental evidence also indicates that joint flexibility should be accounted for in both modeling and control of manipulators [1 5 14]. The difficulty in controlling flexible members promoted most robots to be designed mechanically stiff and rigid, in spite of the fact that joint compliance is a desired feature, as damage can be minimized due to accidental collisions or when large contact forces occur during compliant motions.

Some special problems arise in Flexibly Jointed Robots (FJR) which limit the use of some well established traditionally known in rigid robot control schemes. For example, it has been popular to "globally linearize" the nonlinear rigid robot dynamics because of the special structure of the equations and because the control τ (torque/force) provides an independent input for each degree of freedom. In FJR, there is no independent input for each degree of freedom [5 14], therefore the idea of global linearization of the nonlinear dynamic equations does not work. Furthermore, the desired actuator trajectory in a flexible joint robot is dependent not only on the desired kinematic trajectory of the link but also on the link dynamics. Unfortunately, link dynamic parameters, especially the load dependent ones, are unknown in most cases, as a result the desired actuator trajectory is also unknown. To overcome this difficulty, a number of control schemes require the use of link acceleration and link jerk feedback [2 4], even though it is difficult to measure these signals in real life systems. Another control scheme is based on the concept of reduced order system and requires the restriction of the system to a suitable integral manifold in the state space, and exact knowledge of the parameters [13].

3. Manipulator Models

Experimental investigations of industrial robots with harmonic drive transmission and other forms of gearing indicate that joint flexibility contributes significantly to the overall dynamics of the system [1 13 14]. The dynamic equations of the flexible joint robots are given as :

$$\tau = D_m \ddot{q}_m + B_m \dot{q}_m + K_s(q_m - q) \tag{1}$$

$$0 = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + K_s(q - q_m) \tag{2}$$

where, an n-link manipulator becomes a 2n-degrees of freedom system [1 2 4 5]:

D_m : Diagonal motor inertia matrix ϵ $\mathbb{R}^{n \times n}$
B_m : Diagonal motor damping matrix ϵ $\mathbb{R}^{n \times n}$
K_s : Diagonal drive shaft stiffness matrix ϵ $\mathbb{R}^{n \times n}$
q_m : Vector of sensed motor angles ϵ $\mathbb{R}^{n \times 1}$
$D(q)$: Link inertia matrix ϵ $\mathbb{R}^{n \times n}$
$C(q,\dot{q})$: Centrifugal and Coriolis terms matrix ϵ $\mathbb{R}^{n \times n}$
$g(q)$: Gravitational vector term ϵ $\mathbb{R}^{n \times 1}$
q : Vector of link joint angles ϵ $\mathbb{R}^{n \times 1}$

The above model was derived under a few assumptions, for a complete analysis of this dynamical model, we refer the reader to [14 18].

Matrices D_m, B_m, K_s, are positive definite matrices. Further, $D(q)$ is symmetric, positive definite and both $D(q)$ and $D^{-1}(q)$ are both bounded as function of q [14]. When K_s tends toward infinity, the robot is considered to have rigid joints ($q=q_m$). The dynamic equation which represents the rigidly jointed robot is:

$$\tau = [D(q)+D_m]\ddot{q} + [C(q,\dot{q})+B_m]\dot{q} + g(q) \tag{3}$$

Some properties of the rigid model concerning the inertia matrix, Coriolis and centrifugal force matrix were discussed by Koditschek [7]. Those properties remain valid for the FJR [5 18]. The first most important property shows that $D(q)$ and $C(q,\dot{q})$ are not independent, but the matrix $(\dot{D}-2C)$ is skew symmetric for a suitable definition of the matrix C. However the quantity $\dot{q}^T(\dot{D}-2C)\dot{q}$ is always zero regardless of the choice of C [8]. The second property confirms that the individual terms of the right hand side of equation (2), excluding the $K_s(q-q_m)$ term, can be represented by a linear relationship between a suitably selected set of unknown manipulator and load parameters [5 12 14], in other words equation (2) could be rewritten as:

$$0 = Y(\ddot{q},\dot{q},q)\,P + K_s\,(q - q_m) \tag{4}$$

where, $Y(\ddot{q},\dot{q},q)$ ϵ $\mathbb{R}^{n \times r}$, is called the regressor matrix of known functions, and P ϵ $\mathbb{R}^{r \times 1}$ is a vector of r unknown parameters.

4. Trajectory Model

Let $q_d(t)$ ϵ C^4 denote a desired link trajectory which is continuously differentiable up to the fourth order. The set of desired motor trajectory can be derived using equation (4). The diagonal stiffness matrix, K_s ϵ $\mathbb{R}^{n \times n}$, can be written as K_s = Diag [k_{si}] where, $k_{si} > 0$, for $i = 1,2,...,n$, represents the spring constant of the i^{th} drive shaft. Resulting in a matrix K_s which is invertible and positive definite. We assume the link parameters and the load handled by the end effector to be time invariant, i.e.

$$P = \text{Constant vector, thus, } \dot{P} = \ddot{P} = 0 \tag{5}$$

where, $(\dot{\ })$ stands for time derivative of ( ). The above assumption is valid in a large class of applications. The estimated desired motor trajectory may now be derived from (4), using $(\tilde{\ })$ to denote an estimated quantity, we have:

$$\tilde{q}_{md}(t) = K_s^{-1} Y(\ddot{q}_d,\dot{q}_d,q_d)\tilde{P} + q_d(t) \tag{6}$$

$$\dot{\tilde{q}}_{md}\ (t) = K_s^{-1}[\dot{Y}(\dddot{q}_d,\ddot{q}_d,\dot{q}_d,q_d)\tilde{P} + Y(\ddot{q}_d,\dot{q}_d,q_d)\dot{\tilde{P}}] + \dot{q}_d(t) \tag{7}$$

$$\begin{aligned} \ddot{\tilde{q}}_{md}\ (t) = K_s^{-1}[&\ddot{Y}(\ddddot{q}_d,\dddot{q}_d,\ddot{q}_d,\dot{q}_d,q_d)\tilde{P} + 2\dot{Y}(\dddot{q}_d,\ddot{q}_d,\dot{q}_d,q_d)\dot{\tilde{P}} \\ &+ Y(\ddot{q}_d,\dot{q}_d,q_d)\ddot{\tilde{P}}] + \ddot{q}_d(t) \end{aligned} \tag{8}$$

The subscript "d" is used to denote the desired trajectory. Notice that the estimated desired motor trajectory $[\tilde{q}_{md}(t), \dot{\tilde{q}}_{md}(t)$ and $\ddot{\tilde{q}}_{md}(t)]$ is dependent on the desired link trajectory $[q_d(t), \dot{q}_d(t), \ddot{q}_d(t), \dddot{q}_d(t), \ddddot{q}_d(t)]$, on the estimated parameters $(\tilde{P}, \dot{\tilde{P}}, \ddot{\tilde{P}})$, and on the link dynamic structure $(Y_d, \dot{Y}_d, \ddot{Y}_d)$. This makes it difficult to design a control law which utilizes the desired motor acceleration.

5. Control and Adaptation Law Design

We now state a lemma (a special case of Barbalat's) that will be used in the proof of theorem 1.

Lemma 1:

Let $f(\cdot)$ be a scalar function. If

(1) f is lower bounded, i.e., $\exists C, \forall t \geq 0, f(t) \geq C$;

(2) f has a uniformly continuous, non-positive derivative $\dot{f}$;

then $f(t) \rightarrow 0$ as $t \rightarrow \infty$ [15].

As the dynamic parameters of the arm are unknown and assumed to be time invariant, we can define the parameter error vector as $e_p = \tilde{P} - P$, where $\tilde{P}$ is the estimated parameter vector. Notice $\dot{e}_p = \dot{\tilde{P}}$, as $\dot{P} = 0$. We can define the following motor error as $\tilde{e}_m = (\tilde{q}_{md} - q_m)$, similar terms for $\dot{\tilde{e}}_m$ and $\ddot{\tilde{e}}_m$ can be defined.

Further, let us introduce the following variables to be used in the derivation of the suggested controller:

$$s = \dot{e} + \Lambda e \quad ; \quad s_m = \dot{\tilde{e}}_m + \Lambda_m \tilde{e}_m \tag{9}$$

and

$$W(\dot{e}, e, \ddot{q}_d, \dot{q}_d, \dot{q}, q)P = D(q)[\ddot{q}_d + \Lambda\dot{e}] + C(\dot{q}, q)[\dot{q}_d + \Lambda e] + g(q) \tag{10}$$

where, s and $s_m \in \mathbb{R}^n$ are sliding auxiliary variables, Λ and $\Lambda_m \in \mathbb{R}^{n \times n}$ are positive diagonal matrices, $W \in \mathbb{R}^{n \times r}$ is a regressor matrix. Furthermore, let

$$\beta(\dot{\tilde{P}}, \tilde{P}, \dot{q}_m, q_m, q) = D_m[K_s^{-1}\ddot{Y}_d\tilde{P} + \ddot{q}_d + \Lambda_m \dot{\tilde{e}}_m] + B_m \dot{q}_m \tag{11}$$

where, $\beta \in \mathbb{R}^n$ is defined to simplify the proof of theorem 1. We now state the main theorem of this paper using the above definitions and notations.

Theorem 1 (Controller Derivation): The following direct adaptive controller is a Stable Adaptive Law for the given parametric model (Equations 1 and 2).

$$\tau = \begin{cases} \tau_1 & \text{if } \|s_m\|^2 \neq 0 \\ \tau_2 & \text{if } \|s_m\|^2 = 0. \end{cases} \tag{12}$$

where

$$\tau_1 = \beta(\dot{\tilde{P}}, \tilde{P}, \dot{q}_m, q_m, q) + K_{dm} s_m + K_s(q_m - q) + \frac{s_m}{\|s_m\|^2}\{s^t[W\tilde{P} + K_s(q - q_m) + K_d s]\} \tag{13}$$

$$\tau_2 = W(\dot{e}, e, \ddot{q}_d, \dot{q}_d, \dot{q}, q)\tilde{P} + \beta(\dot{\tilde{P}}, \tilde{P}, \dot{q}_m, q_m, q) + K_d s \tag{14}$$

with K_d, and K_{dm} are some diagonal positive gain matrices $\in \mathbb{R}^{n \times n}$. At all times, we have one adaptation law suggested by Theorem 2. The above suggested adaptive controller guarantees zero position and velocity tracking errors, as well as bounded parameter error for a subclass of desired link trajectories described in Theorem 2.

Proof of Theorem 1:

Let us differentiate the following Lyapunov function candidate V(t),

$$V(t) = \tfrac{1}{2}s_m^t D_m s_m + \tfrac{1}{2}s^t D(q)s + \tfrac{1}{2}e^t Re + \tfrac{1}{2}\tilde{e}_m^t R_m \tilde{e}_m + \tfrac{1}{2}e_p^t M e_p \tag{15}$$

where, R and $R_m \in \mathrm{IR}^{n \times n}$ are some positive matrices to be chosen toward the conclusion of the proof [16].

$$\begin{aligned}\dot{V}(t) &= s_m^t D_m(\ddot{\tilde{e}}_m + \Lambda_m \dot{\tilde{e}}_m) + e^t R \dot{e} + \tilde{e}_m^t R_m \dot{\tilde{e}}_m \\ &\quad + s^t[D(q)(\ddot{e} + \Lambda \dot{e}) + \tfrac{1}{2}\dot{D}(\dot{e} + \Lambda e)] + e_p^t M \dot{e}_p - s^t(\tfrac{1}{2}\dot{D} - C)s \end{aligned} \tag{16}$$

In order to simplify equation (16), we have subtracted the term $s^t(\frac{1}{2}\dot{D} - C)s = 0$. Simplifying equation (16) and substituting the dynamic equation of the FJR given by (1) and (2), we have

$$\begin{aligned}\dot{V}(t) &= s_m^t [D_m(\ddot{\tilde{q}}_{md} + \Lambda_m \dot{\tilde{e}}_m) - D_m \ddot{q}_m] + \tilde{e}_m^t R_m \dot{\tilde{e}}_m \\ &\quad + s^t\{D(q)(\ddot{q}_d + \Lambda \dot{e}) + C(\dot{q}_d + \Lambda e) - [D(q)\ddot{q} + C\dot{q} + K_s(q - q_m) + g(q)] \\ &\quad + K_s(q - q_m) + g(q)\} + e_p^t M \dot{e}_p + e^t R \dot{e} \\ &= s_m^t [D_m(\ddot{\tilde{q}}_{md} + \Lambda_m \dot{\tilde{e}}_m) + K_s(q_m - q) + B_m \dot{q}_m - \tau] \\ &\quad + s^t[WP + K_s(q - q_m)] + e_p^t M \dot{e}_p + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} \end{aligned} \tag{17}$$

Case a: Let us assume that $\|s_m\|^2 \neq 0$ in this case, hence we can substitute the derived expression for $\ddot{\tilde{q}}_{md}$ from equation (8).

$$\begin{aligned}\dot{V}(t) &= s_m^t [D_m(K_s^{-1} \ddot{Y}_d \tilde{P} + \ddot{q}_d + \Lambda_m \dot{\tilde{e}}_m) + K_s(q_m - q) + B_m \dot{q}_m - \tau] \\ &\quad + s^t[W\tilde{P} + K_s(q - q_m)] + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} \\ &\quad + s_m^t D_m K_s^{-1} [2\dot{Y}_d \dot{\tilde{P}} + Y_d \ddot{\tilde{P}}] + e_p^t [M \dot{e}_p - W^t s]. \end{aligned} \tag{18}$$

using the definition of $\beta(\dot{\tilde{P}}, \tilde{P}, \dot{q}_m, q_m, q)$ from equation (11), the above expression could be rewritten as:

$$\dot{V}(t) = s_m^t [\beta_1(\dot{\tilde{P}}, \tilde{P}, \dot{q}_m, q_m, q) + K_s(q_m - q) - \tau] + s^t[W\tilde{P} + K_s(q - q_m)] + \tilde{e}^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} \tag{19}$$

the parameters are estimated from the below set of equations (see Theorem 2 for details):

$$\begin{cases} Y_d \ddot{\tilde{P}} + 2\dot{Y}_d \dot{\tilde{P}} = 0 \\ M \dot{\tilde{P}} = W^t s \end{cases} \tag{20}$$

and therefore a substitution of τ_1 above will yield

$$\dot{V}(t) = -s^t K_d s - s_m^t K_{dm} s_m + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} \tag{21}$$

Case b: When we have $\|s_m\|^2 = 0$, we suggest monitoring the derivative of the following function (see "Conclusion of the Proof" for a complete analysis).

$$F(t) = \tfrac{1}{2} s^t D(q) s + s_m^t D_m s + \tfrac{1}{2} e^t R e + \tfrac{1}{2} \tilde{e}_m^t R_m \tilde{e}_m + \tfrac{1}{2} e_p^t M e_p \tag{22}$$

taking the time derivative of the above function, and using the combined dynamics of links and motors in equations (1) and (2):

$$\begin{aligned}\dot{F}(t) &= s^t[D(q)(\ddot{e} + \Lambda \dot{e}) + \tfrac{1}{2}\dot{D}(\dot{e} + \Lambda e) + D_m(\ddot{\tilde{e}}_m + \Lambda_m \dot{\tilde{e}}_m)] + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} + e_p^t M \dot{e}_p \\ &= s^t[D(q)(\ddot{q}_d + \Lambda \dot{e}) + C(q, \dot{q})(\dot{q}_d + \Lambda e) + D_m(\ddot{\tilde{q}}_{md} + \Lambda_m \dot{\tilde{e}}_m) \\ &\quad - \tau + B_m \dot{q}_m + g(q)] + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} + e_p^t M \dot{e}_p \end{aligned} \tag{23}$$

Therefore, a substitution of τ_2 in the above expression yields:

$$\dot{F}(t) = -s^t K_d s + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} \tag{24}$$

where the assumption about the estimated parameters is the same one defined earlier (20). On the other hand $\|s_m\|^2 = \dot{0}$ in this case, which allows us to add a quadratic term in s_m in the last expression of $\dot{F}(t)$.

$$\dot{F}(t) \leq -s^t K_d s - s_m^t K_{dm} s_m + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} \tag{25}$$

Conclusion of the Proof: We now show that V(t) (the Lyapunov candidate) is non-increasing at all times. The main concern is over the behavior of V(t) in case (b) (i.e., when $\|s_m\|^2 = 0$). Notice that

$$V(t) = F(t) + \tfrac{1}{2} s_m^t D_m s_m - s^t D_m s_m \tag{26}$$

we can drop the last two terms in the above equation, since in this case (b), $\|s_m\|^2 = 0$ iff $s_m = 0$. Taking the time derivative of the above expression yields

$$\dot{V}(t) \leq -s^t K_d s - s_m^t K_{dm} s_m + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} \tag{27}$$

Hence, we have shown that V(t) has the same upper bound at all times, in fact whether $\tau = \tau_1$ or τ_2, we have:

$$\dot{V}(t) \leq -s^t K_d s - s_m^t K_{dm} s_m + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} \tag{28}$$

Invoking our definition of the auxiliary variables s and s_m and a proper substitution in the upper bound of $\dot{V}(t)$ yields,

$$\begin{aligned} \dot{V}(t) \leq & -\dot{e}^t K_d \dot{e} - e^t \Lambda^t K_d \Lambda e - 2e^t \Lambda^t K_d \dot{e} \\ & - \dot{\tilde{e}}_m^t K_{dm} \dot{\tilde{e}}_m - \tilde{e}_m^t \Lambda_m^t K_{dm} \Lambda_m \tilde{e}_m - 2\tilde{e}_m^t \Lambda_m^t K_{dm} \dot{\tilde{e}}_m \\ & + \tilde{e}_m^t R_m \dot{\tilde{e}}_m + e^t R \dot{e} \end{aligned} \tag{29}$$

If we choose, $R = 2\Lambda^t K_d$, and $R_m = 2\Lambda_m^t K_{dm}$, then we can express $\dot{V}(t)$ as :

$$\dot{V}(t) \leq -\dot{e}^t K_d \dot{e} - e^t (\Lambda^t K_d \Lambda) e - \dot{\tilde{e}}_m^t K_{dm} \dot{\tilde{e}}_m - \tilde{e}_m^t (\Lambda_m^t K_{dm} \Lambda_m) \tilde{e}_m \leq 0 \tag{30}$$

From this last expression of $\dot{V}(t)$, we can conclude that since V(t) is a lower bounded function and V(t) is uniformly continuous, and negative semidefinite, therefore by lemma 1, we have:

$$\dot{V}(t) \rightarrow 0 \text{ as } t \rightarrow \infty \tag{31}$$

$$\text{i.e., } [\dot{e}(t),\ e(t),\ \dot{\tilde{e}}_m(t), \text{ and } \tilde{e}_m(t)] \rightarrow 0 \text{ as } t \rightarrow \infty. \tag{32}$$

###

and bounded parameter error.

Remark 1.1: For practical purposes, the control torque switching condition may be relaxed to avoid large torques, such that $\tau := \tau_2$ if $\|s_m\|^2 \leq \epsilon$. For some significantly small scalar ϵ.

Corollary 1.1: In the non-adaptive case (when parameters are known) the system (Equations 1 and 2) subjected to the controller of Theorem 1 (Equations 12, 13, and 14) is asymptotically stable. This is clear, since V(t) will become negative definite instead of being negative semi-definite in the adaptive case.

Theorem 2 (Adaptation law):

The above controller of Theorem 1, and adaptation law below, is globally stable if rank $[Y_d Y_d^t] = n$, and

$$\exists \lambda(t_0) \mid \det[\lambda(t_0) I + 2Y_d^t (Y_d Y_d^t)^{-1} \dot{Y}_d(t_0)] \neq 0 \tag{33}$$

In Theorem 1 the parameter adaptation scheme was given by equation (20). Since the regressor Y_d is usually rectangular, $Y_d \in \mathbb{R}^{n \times r}$ where r (the number of unknown parameters) is in general larger than n (the number of links). Therefore, if we restrict the class of tractable desired link trajectories to those implying full (column) rank of

Y_d, then $Y_d(t)Y_d^t(t) \in \mathbb{R}^{n\times n}$ is nonsingular $\forall t \geq t_0$. Hence, a right inverse of Y_d ($Y_d^{-R} \in \mathbb{R}^{r\times n}$) exists, namely the *Moore-Penrose pseudo (right) inverse.* Now, we can assign the following variables. Let $x_1(t)=\tilde{P}(t)$, $x_2(t)=\dot{\tilde{P}}(t)$, and $x_3(t)=M^{-1}W^t(t)s(t) \in \mathbb{R}^r$, the equivalent constrained system (20) could be rewritten as:

$$\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & I & 0 \\ 0 & -2Y_d^{-R}\dot{Y}_d & 0 \\ 0 & -I & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{34}$$

where, I is the rxr identity matrix. This is a time varying linear singular system (or descriptor system). The above system could be formulated as follows, denoting the coefficient matrices by A and $B(t) \in \mathbb{R}^{3r\times 3r}$.

$$A\dot{x}(t) = B(t)x(t) \tag{35}$$

Note that, A is singular at all time, and $x_3(t)$ is a known function of time which could be computed on line. We can solve this system for $\tilde{P}(t)$ and $\dot{\tilde{P}}(t)$, given an initial set of conditions $\tilde{P}(0)$ and $\dot{\tilde{P}}(0)$. The solution of this singular time varying linear system has been discussed extensively in the literature, Campbell in [19] and a large group of other researchers have studied this problem and have suggested several solutions [20]. The existence of a numerical solution for the above "descriptor" system is conditioned by the following. The existence of a $\lambda \in \mathbb{R}$, which may depend on the initial time t_0, such that $[\lambda A(t_0) - B(t_0)]$ is nonsingular. Equivalently, for the above system we have, this condition on the solution existence is rewritten as: $\exists \lambda$ such that the below matrix is nonsingular:

$$[\lambda A(t_0) - B(t_0)] = \begin{bmatrix} \lambda I & -I & 0 \\ 0 & \lambda I + 2Y_d^{-R}\dot{Y}_d(t_0) & 0 \\ 0 & I & -I \end{bmatrix} \tag{36}$$

The above matrix is invertible when $[\lambda I + 2Y_d^{-R}(t_0)\dot{Y}_d(t_0)]$ is nonsingular. Therefore, we need to find $\lambda(t_0)$ that satisfies this lower order condition.

###

Remark 2.1: For a number of cases we simulated, a valid $\lambda(t_0)$ was equal to one. This $\lambda(t_0)$ is easy to find, since it is only required for the initial time t_0 and depends on the given desired trajectory.

Remark 2.2: Let us mention that the solution to the above singular system may be subject to some impulsive responses due to inconsistency in the equations composing the system. However, this undesired part of the response is transient and could be suppressed by a proper choice of the initial conditions. In particular, an initial set of conditions or estimates $(\tilde{P}(0), \dot{\tilde{P}}(0))$ can be selected such that it satisfies

$$\dot{\tilde{P}}(0) = M^{-1}W^t(0)s(0) \tag{37}$$

this will eliminate the impulsive parts of the solution.

6. Simulation Results for a Two-Link Planar FJR.

We now describe the computer simulation for a two-link planar manipulator with revolute joints. The linkage are composed of two identically uniform beams which are infinitely rigid, with actuators mounted at the joints (see Figure 1). We assume that the load carried by the end-effector is a part of the second link. We selected a sampling period of 10×10^{-3} seconds corresponding to a servo rate of 100 Hz. The motor known parameters are: $K_{s1}=K_{s2} = 50$ N m rad^{-1}; $D_{m1}=D_{m2}= .05$ kg m^2; $B_{m1}=B_{m2}= .05$ N m sec rad^{-1}. On the other hand the true values of the link unknown parameters are: $P_1 = 1.66$; $P_2 = 0.42$; $P_3 = 0.63$; $P_4 = 3.75$; $P_5 = 1.25$. Where each link extended 0.5 m, and weighted 5 kg. In the below simulations, the desired trajectory is given by:

$$q_{1d}(t) = [-\frac{\pi}{2} + 0.3\sin(\pi t)]\text{rad}. \tag{38}$$

$$q_{2d}(t) = [-0.3 + 0.3\cos(\pi t)]\text{rad}. \tag{39}$$

Case #1: n this case, we present the "Non-Adaptive" scheme suggested by Theorem 1 with known parameters. The following gain matrices were used.

$$\Lambda=\Lambda_m=\begin{bmatrix}3 & 0\\ 0 & 3\end{bmatrix} \quad K_d=K_{dm}=\begin{bmatrix}5 & 0\\ 0 & 5\end{bmatrix} \tag{40}$$

We present the tracking errors of the links in Figure 2a, and the tracking errors of the motors in Figure 2b.

Case #2: In order to show the effectiveness of the adaptive control scheme given by Theorems 1 and 2, with the approximate solution of the singular system (34), we applied our two level control scheme to the FJR. The responses are given in Figures (3a, 3b, and 3c). Figure 3a shows the responses for the links, while Figure 3b shows the responses of the motors, and Figure 3c shows the estimates of the parameters. We can see that the motor and the link tracking errors go to zero. The parameters also do not diverge, although they do not converge to their exact actual values. In the adaptation law, the solution of the singular system was truncated to simplify the simulation. The controller gain matrices given in equation (40) were used for this case. In addition, the adaptation gain matrix $M = 2\ I_5$, where I_5 is the 5x5 identity matrix.

9. Conclusion

In this paper we have presented a direct adaptive scheme to control flexible joint robots. Adaptive controller for the FJR was derived using Lyapunov's second method. Global asymptotic stability (in the Non-Adaptive case) was verified through simulations (case #1). Global stability of the adaptive controller was proven in theorems 1 and 2. The control scheme is significantly more complex than the rigid robot counterpart. The parameter update law has to be implemented numerically as it is derived from a singular system of equations. The complexity of the control scheme arises from the dynamic structure of the FJR and the lack of direct input to the links. Several improvements to this control mechanism are currently being considered. These improvements can include the insertion of a boundary layer to prevent discontinuous switching (controller chattering). In addition, the derived controller can be made robust by modifying the parameter update law to prevent parameters divergence in the presence of disturbances. We note that the existence of Y_d^{-R} was possible in many cases of simulations. We believe for all robots, trajectories satisfying this condition should be feasible. The physical insights to this condition needs to be clarified.

10. Acknowledgement

We would like to thank Professors Martin Corless, Stanislaw Żak, and Mark Spong for discussions on the area of flexibly jointed manipulators and on Lyapunov stability analysis.

REFERENCES

[1] S. Ahmad, G. Widmann, "Control of Robots with Flexible Joints," *Proceedings of 1987 IEEE International Conference on Robotics and Automation,* Raleigh, NC, pp. 1561-1566, March 1987.

[2] A. De-Luca, "Dynamic Control of Robots with Joint Elasticity," *Proceedings of 1988 IEEE International Conference on Robotics and Automation,* Philadelphia, PA, pp. 152-158, April 1988.

[3] A. Ficola, R. Marino, S. Nicosia, "A singular perturbation approach to the control of elastic robots," *Proceedings of the 21st Ann. Allerton Conf. Communications, Control, and Computing,* University of Illinois, 1983.

[4] L. Fu, K. Chen, "Nonlinear Adaptive Motion Control for a Manipulator with Flexible Joints," *Proceedings of 1989 IEEE International Conference on Robotics and Automation,* Phoenix, Az, pp. 1201-1206, May 1989.

[5] F. Ghorbel, M. Spong and J. Hung, "Adaptive Control of Flexible Joint Manipulators," *Proceedings of 1989 IEEE International Conference on Robotics and Automation,* Phoenix, Az, pp. 1188-1193, May 1989.

[6] P. Ioannou and P. Kokotovic, "Instability Analysis and Improvement of Robustness of Adaptive Control," *Automatica,* Vol. 20, No. 5, pp. 583-594, 1984

[7] D. Koditschek, "Lyapunov Analysis of Robot Motion," *Tutorial Workshop of 1987 IEEE International Conference on Robotics and Automation,* Raleigh, NC, pp. 1B-3-1 to 1B-3-56, March 1987.

[8] R. Ortega, M. Spong, "Adaptive Motion Control of Rigid Robots: A Tutorial," *Proceedings of the 27th Conference on Decision and Control,* Austin, Texas, Dec. 1988.

[9] R. Paul, *Robot Manipulators: Mathematics, Programming and Control,* MIT Press, 1981

[10] E. Rivin, "Effective Rigidity of Robot Structures: Analysis and Enhancement," *Proceedings of the 1985 American Control Conference,* Boston, MA, pp. 381-382, June 1985.

[11] S. Sastry, P. Hsu, M. Bodson and B. Paden, "Adaptive Identification and Control for Manipulators without Using Joint Accelerations," *Proceedings of 1987 IEEE International Conference on Robotics and Automation,* Raleigh, NC, pp. 1210-1215, March 1987.

[12] J. Slotine and W. Li, "Adaptive Manipulator Control: A Case Study," *IEEE Trans. Automatic Control,* Vol. 33, No. 11, pp. 995-1003, Nov 1988

[13] M. Spong, K. Khorasani and P.Kokotovic, "An Integral Manifold Approach to the Feedback Control of Flexible Joint Robots," *IEEE J. of Robotics and Automat.,* Vol. RA-3, No. 4, pp. 291-300, Aug 1987

[14] M. Spong, "Modeling and Control of Elastic Joint Robots," *Transactions of the ASME,* Vol 109, pp. 310-319, Dec 1987.

[15] J. Slotine and W. Li, "Composite Adaptive Control of Robot Manipulators," *Automatica,* Vol. 25, No. 4, pp. 509-519, 1989

[16] M. Spong, R. Ortega, R. Kelly, "Comments on: Adaptive Manipulator Control: A Case Study," *IEEE Trans. on Automatic Control,* Vol. 35, No. 6, pp. 761-762, June 1990.

[17] M. Corless,"Control of Uncertain Mechanical Systems With Robustness in the Presence of Unmodelled Flexibilities," *Proc. of the Third Bellman Continuum,* Sophia-Antipolis, France, 1988.

[18] M. Spong,"Control of Flexible Joint Robots: A Survey," *Coordinated Science Laboratory,* University of Illinois At Urbana-Champaign, R-2203, February 1990.

[19] S. Campbell, "One Canonical Form for Higher Index Linear Time-Varying Singular Systems," *Circuits Systems and Signal Processing,* Vol. 2, No. 3, pp. 311-326, 1983.

[20] J. Mills, A. Goldenberg, "Force and Position Control of Manipulators During Constrained Motion Tasks," *IEEE Trans. on Robotics and Automation,* Vol. 5, No. 1, pp. 30-46, February 1989.

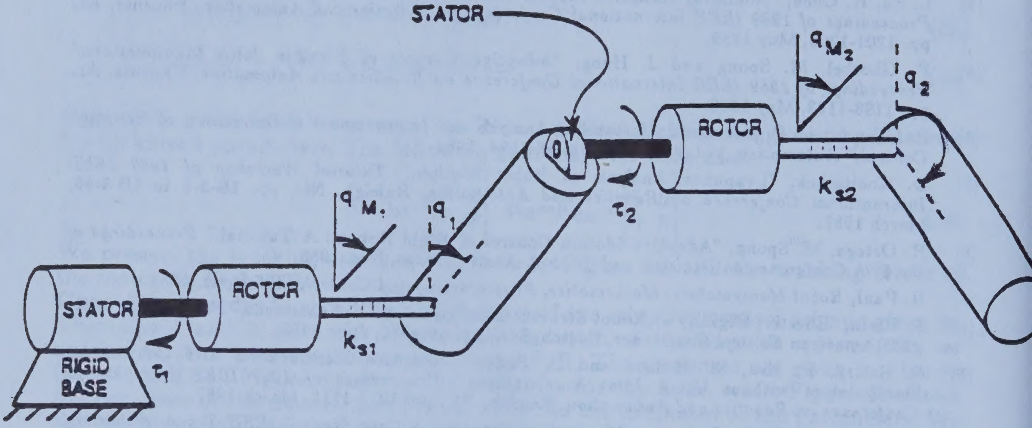

STATOR
ROTOR
RIGID BASE
τ1
k_s1
q_M1
q_1
STATOR
ROTOR
q_M2
q_2
τ2
k_s2

Figure 1. Conceptual Diagram of Two Link Compliant Manipulator

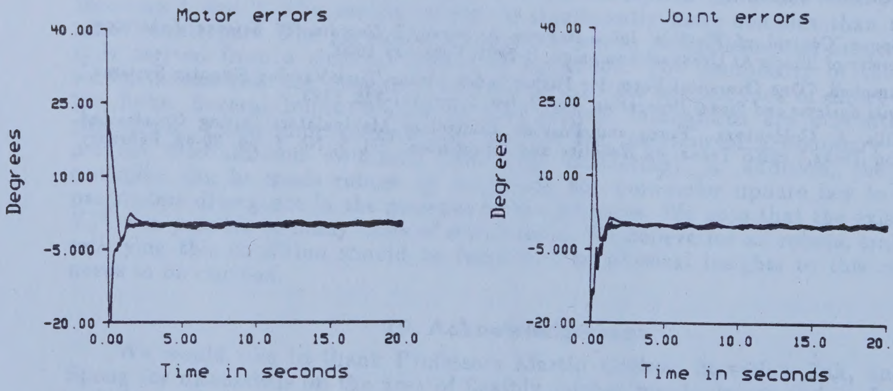


Figure 2a

Figure 2b

Case #1: Non-Adaptive controller response for FJR.

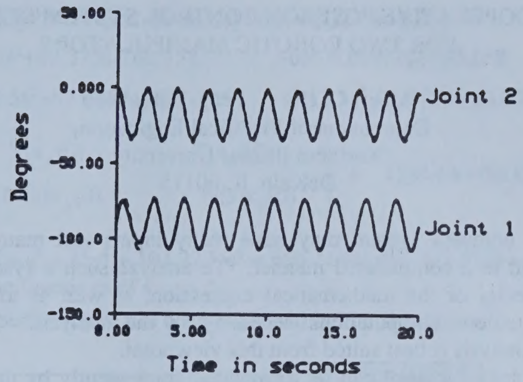


Figure 3a

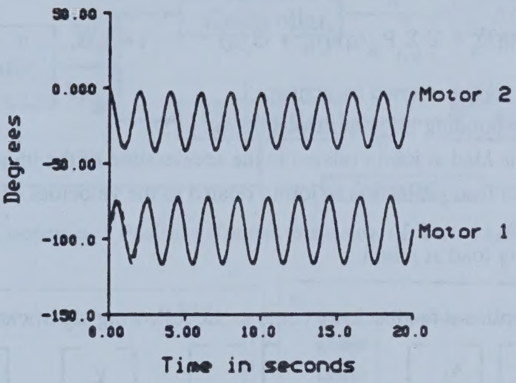


Figure 3b

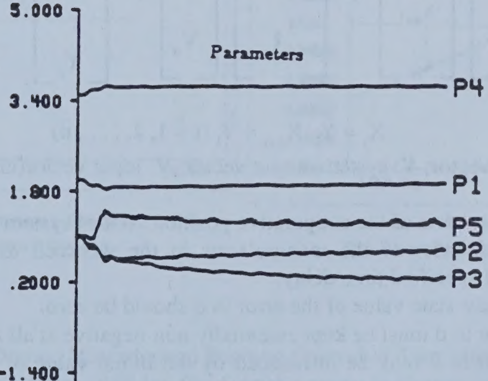


Figure 3c

Case #2: Adaptive controller response for FJR.

COOPERATIVE POSITION CONTROL SYSTEM DESIGN FOR TWO ROBOTIC MANIPULATORS

Albert C. Hu Peng-Yung Woo
Department of Electrical Engineering
Northern Illinois University
DeKalb, IL 60115

A modern complex system may have many inputs and many outputs, and these may be interrelated in a complicated manner. To analyze such a system, it is essential to reduce the complexity of the mathematical expression, as well as to resort to computers for most of the tedious computations necessary in the analysis. The state space approach to system analysis is best suited from this viewpoint.

The dynamics of a robot can be formulated conveniently by the Lagrangian formulation. The bahavior of a rigid n-link maniupulator results in an equation of the form:

$$\tau_i = \sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} P_{ijk}(q)\dot{q}_j\dot{q}_k + G_i(q) \qquad i = 1, 2, \ldots n$$

where τ_i = joint torque exerted by actuator i.

q_i = corresponding ith joint displacement.

$D_{ij}(q)$ = inertia load at joint i related to the acceleration of the jth joint.

$P_{ijk}(q)$ = inertia load projection to joint i related to the velocities of the jth and the kth joints.

$G_i(q)$ = gravity load at joint i.

By using nonlinear feedback,we come to the following significant conclusion:

$$\begin{bmatrix} \dot{X}_1 \\ \vdots \\ \dot{X}_n \\ \dot{X}_{n+1} \\ \vdots \\ \dot{X}_{2n} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_n \\ X_{n+1} \\ \vdots \\ X_{2n} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} ; \quad \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} I \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$$

Where

$$X_i = Y_i,\ X_{n+i} = \dot{Y}_i \ (i = 1, 2, \ldots, n)$$

X: state variable vector; Y: system output vector; V: input vector(all in world coordinates)

The design criteria of the cooperative position control system are as following:

1. The responses of the manipulators in the workcell to their respective inputs must have identical specified time delay.

2. The steady-state value of the error in d should be zero.

3. The error in d must be kept essentially non-negative at all times.

4. The error in d may be influenced by the initial value of d, but not by the initial values of the absolute positions of X_A and X_B.

5. The shape of X_{Ad}, the desired trajectory of the end-effector holding the workpiece, should have no effect on the shape of the trajectory of d.

We use the improved I-PD control structure and the Bessel prototype to locate the poles and the zeros of the system output to meet some requirements suggested for the system controller design. Also, the appropriate input signals have been generated by a design of the coordinator to make the system meet other suggested requirements [1-3].

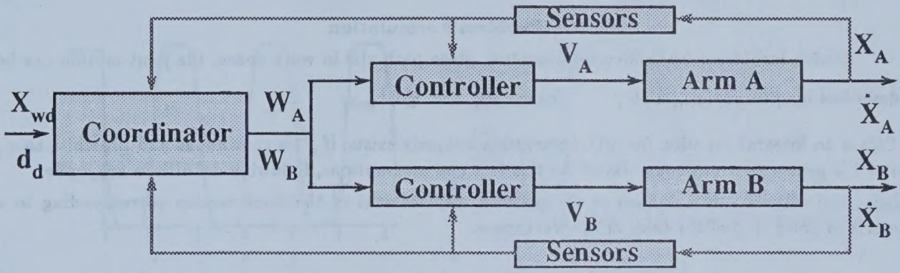
$$\mathbf{Y(s)=C[sI-(A+BK)]^{-1}BW+C[sI-(A+BK)]^{-1}x(0)+C[sI-(A+BK)]^{-1}r}$$

$$=\frac{1}{\Delta}\begin{bmatrix} s^3+7.5s^2+42.375s-260.158 & -6s^2-40.5977s+260.158 \\ s^3+13.5s^2+82.9727s+531.7054 & -s^3-13.5s^2-82.9727s-531.7054 \end{bmatrix}\begin{bmatrix} W_A{}^x(s) \\ W_B{}^x(s) \end{bmatrix}$$

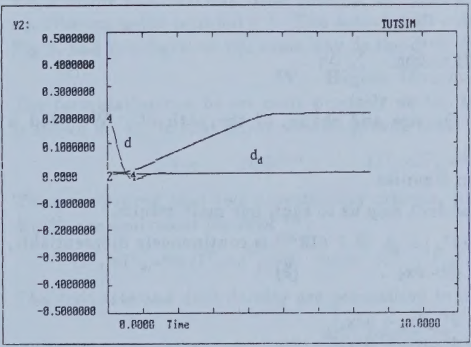
$$+\frac{1}{\Delta s}\begin{bmatrix} T_1x_A(0) & T_2x_B(0) \\ M(s)x_A(0) & N(s)x_B(0) \end{bmatrix} + \mathbf{C[sI-(A+BK)]^{-1}r}$$

Where $M(s) = N(s) = s^5 + 15s^4 + 103.2227s^3 + 656.16445s^2 + 797.5581s$
T_1, T_2 are the functions of s.

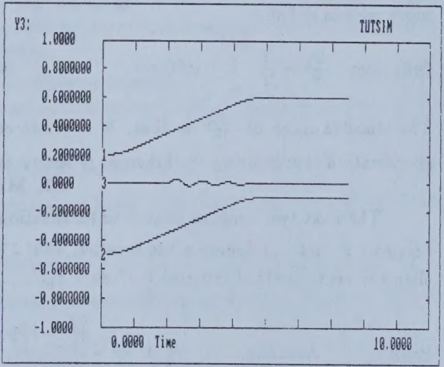
The system performance has been evaluated by the simulation.



Cooperative Position Control Structure of Two Manipulators.



Desired and Actual Trajectory of the Distance Between the Two End-Effectors in R Dimension



System Output by Adding Some Kinds of Disturbance to the Actuator on the Manipulator

References:

[1] P. Y. Woo, & N. Dorny, "Coordinated Position Control of Two Manipulators Working Together," Proc. 1988 Int. Conf. on Advances in Comm. and Control, Baton Ronge, Louisiana, Oct. 1988.

[2] P. Y. Woo, "Cooperative Force Control of Two Mutli-Link Manipulators," Proc. 1989 American Control Conf, Pittsburgh, Pennsylvania, June 1989.

[3] P. Y. Woo, & N. Dorny, "Coordinated Control in a Multi-Manipulator Workcell," Advances in Computing and Control. edited by W. A. Porter, S. C. Kak and J. L. Aravena; Springer-Verlag, 1989.

ON THE MEASURE OF REPEATABILITY OF REDUNDANT ROBOTS

Shengwu Luo † and Shaheen Ahmad
Real-Time Robot Control Laboratory
School of Electric Engineering, Purdue University
West Lafayette, IN 47907-0501, USA

I. Abstract

When the end-effector of a redundant manipulator traces a closed path in the workspace, the path in the joint space is not necessarily closed. This phenomenon is known as the non-repeatability of the joint motions. This paper presents the mathematical description and predicts the drift behavior, when the manipulator is controlled by Jacobian pseudo-inverses.

II. Problem Formulation

Under Jacobian pseudo-inverse J^+ control, given path $x(t)$ in work space, the joint motion can be described by $q(t)=\int_0^t J^+(q(t))\,\dot{x}\,dt$, given $x(t)$ and $q(0)\,|_{\ker(K)}$

This is an integral equation for $q(t)$. Its solution uniquely exists, if J^+ is continuous and Lipschitz to q , and $\dot{x}$ is piecewise continuous. Based on this fact and assumptions, following definitions are made.

Definition: "Drift" Δq is defined as the position displacement of the joint motion corresponding to a specified point $x(t_k)$ of the path in the workspace.

$$\Delta q(k)=q(t_k+T)-q(t_k) \triangleq \oint_{\partial D} J^+(q)dx= \int_{t_k}^{t_k+T} J^+(q(t))\dot{x}dt \qquad | \quad x(t_k+kT)=x(t_k) \tag{1}$$

Actually they locate along a fiber, i.e. the kernel of the kinematic map $K : q \rightarrow x$, $x=K(q)$. Furthermore we can define,

Drift rate: $\dfrac{\Delta q}{T} = \dfrac{1}{T} \displaystyle\int_{t_k}^{t_k+T} J^+(q(t))\dot{x}dt$ and drift motion : $\sum^{k} \Delta q$.

The disadvantage of $\frac{\Delta q}{T}$ is that, it depends on the size and shape of the path ∂D. We need a specification representing the inherent property of J^+.

III. Main Results

The next two lemmas related to localization of drift help us to show our main results.

Lemma 1 : Let $[.,.]$ denote a Lie bracket, and $J^+=(J^+_1 \,|\, J^+_2)$. If $J^+ \in \mathbb{R}^{3\times 2}$ is continuously differentiable, then the vector valued integral $\oint_{\partial D} J^+\,dx = \int_D [J^+_1 , J^+_2]dx_1dx_2$. (2)

Proof : Actually, $\oint_{\partial D} J^+\,dx = \begin{bmatrix} \oint J^+_{11}dx_1 + J^+_{12}dx_2 + 0dx_3 \\ \oint J^+_{21}dx_1 + J^+_{22}dx_2 + 0dx_3 \\ \oint J^+_{31}dx_1 + J^+_{32}dx_2 + 0dx_3 \end{bmatrix}$

The Stokes' formula on a surface D with boundary ∂D is of the form, $\oint_{\partial D} f dx = \int_D n^0 \bullet \nabla_x \times f\, dS$, for a continuously differentiable vector field f. Note, in x_1,x_2 plane, the normal $n^0 = \mathbf{k}$ (the z-axis unit vector) and the area element $dS = dx_1dx_2$. For each row we apply Stokes' formula,

$$\oint J^+_{\bullet 1}\,dx_1 + J^+_{\bullet 2}\,dx_2 + 0\,dx_3 = \int_D \mathbf{k}\bullet \begin{bmatrix} \mathbf{i} & \partial/\partial x_1 & J^+_{\bullet 1} \\ \mathbf{j} & \partial/\partial x_2 & J^+_{\bullet 2} \\ \mathbf{k} & \partial/\partial x_3 & 0 \end{bmatrix} dx_1\,dx_2 = \int_D \left(\frac{\partial}{\partial x_1} J^+_{\bullet 2} - \frac{\partial}{\partial x_2} J^+_{\bullet 1} \right) dx_1\,dx_2$$

By chain rule of differentiation, $\left(\dfrac{\partial}{\partial x_1} J^+_{\bullet 2} - \dfrac{\partial}{\partial x_2} J^+_{\bullet 1} \right) = \nabla_q J^+_{\bullet 2} \bullet J^+_1 - \nabla_q J^+_{\bullet 1} \bullet J^+_2$

---

† This work was funded by a David Ross Grant from the Purdue University.

Finally, by Lie bracket notation, they yield $\left(\nabla_q J^+{}_2 \bullet J^+{}_1 - \nabla_q J^+{}_1 \bullet J^+{}_2\right) = [J^+{}_1 , J^+{}_2]$ □

Lemma 2 : [see 4] $[J^+{}_1 , J^+{}_2] = \lim_{\mathrm{diam}D \to 0} \frac{1}{\mathrm{area}(D)} \int_D [J^+{}_1, J^+{}_2] dx_1 dx_2 = \lim_{\mathrm{diam}D \to 0} \frac{1}{\mathrm{area}(D)} \Delta q.$

Now we introduce a qualified measure of drift:

Definition "drift density" : $dd(q) \triangleq t^0 \bullet [J^+{}_1 , J^+{}_2]$, where t^0 = tangent vector along the fiber.

From the above two lemmas, it is easily shown that the drift density is a surface density of the drift (rate), which is projected on the fiber. Here, dd(q)=0, means locally no drift occurs, it serves as a measure of repeatability, that is same as Lie bracket condition[3]. Moreover, this measure of drift dd(q) can be used for predicting the behavior of the drift: limit points exist and motion is numerically stable[2]. Following is an example of using the drift density in phase portrait analysis.

Example : Consider a three-link planar manipulator with unit length and joint angles θ_1, θ_2 and θ_3.

$$\text{drift density} = t^0 \bullet [J^+{}_1, J^+{}_2] = t^0 \bullet [J^T{}_1, J^T{}_2]/\det(JJ^T) = \frac{s_{32}+s_{13}+s_{21}}{(s_{32}^2+s_{13}^2+s_{21}^2)^{3/2}}$$

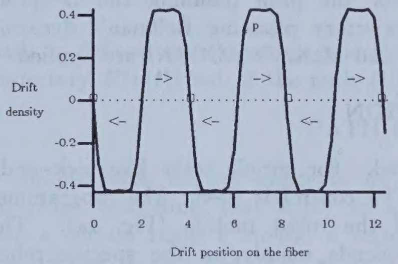


Fig. 1 Phase portrait

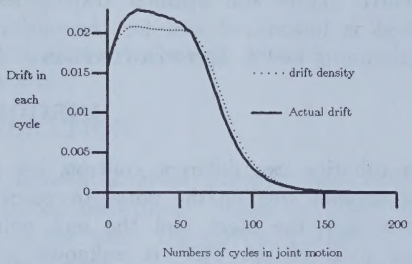


Fig.2 Convergence of the drift

Fig.1 shows that starting from point p the drift is predicted to go up then down to a nearby stable equilibrium point (symbol □). The actual drift calculated by numerical integration of eq(1), shown in Fig.2, and it behaves in the same way as the drift density.

IV. Higher Dimension Generalization

The formulation can be set more precisely on the framework of integration of differential forms. This is shown in [4]. In that paper we also proved that,

For $J \in \mathrm{IR}^{2\times3}$, $(J^T{}_1 \times J^T{}_2) \bullet \nabla \times (J^T{}_1 \times J^T{}_2) = -(J^T{}_1 \times J^T{}_2) \bullet [J^T{}_1, J^T{}_2]$.

Thus, we proved that two repeatability criteria, presented by Klein & Huang[1] and Shamir & Yomdin[3], are equivalent for $J \in \mathrm{IR}^{2\times3}$:

$$(J^T{}_1 \times J^T{}_2) \bullet \nabla \times (J^T{}_1 \times J^T{}_2) = 0 \quad <=> \quad [J^T{}_1 , J^T{}_2] \in \mathrm{Image}(J^T) \quad <=> \quad [J^+{}_1 , J^+{}_2] \in \mathrm{Image}(J^+)$$

The drift rate and drift density are generalized to arbitrary dimensional spaces in [5], where we define

$$\text{drift density} = t^0 \bullet \sum_j \sum_{i<j} [J^+{}_i, J^+{}_j].$$

V. References

[1] C.A. Klein and C.H. Huang, "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators," *IEEE Transactions on System, Man and Cybernetics, V.SMC-13, pp.245-250, 1983.*

[2] C.A. Klein and K.B. Kee, "The Nature of Drift in Pseudo-inverse Control of Kinematically Redundant Manipulators," *IEEE Transactions on Robotics and Automation, V.RA-5. pp231-234, 1989.*

[3] T. Shamir and Y. Yomdin, "Repeatability of Redundant Manipulators: Mathematical Solution of the Problem," *IEEE Trans. on Automatic Control, V.AC-33, pp.1004-1009, 1988.*

[4] S. Luo and S. Ahmad, "Measure of Joint Path Drift for Kinematically Redundant Robots," *summited to 1991 IEEE International Conf. on Robotics and Automation.*

[5] S. Luo and S. Ahmad, "On the Drift Rate in Pseudo-Inverse Control of Redundant Robots" *summited to 1991 American Control Conference.*

Path and Trajectory Generation for Robots

T. L. Tschiesche
Department of Measurement and Control
Prof. Dr.-Ing. H. Schwarz
Faculty of Mechanical Engineering
University of Duisburg
D-4100 Duisburg 1, FRG

Abstract: The execution of demanding manufacturing tasks with robot manipulators requires accurate tracking of a predefined geometric path with the tool center point (TCP) with an optimal velocity profile along the path. This paper describes suitable solutions for the two interdependent problems of the generation of smooth geometric paths and optimal trajectories. For the path planning the B-spline method is introduced and for the optimal trajectory planning Bellman's dynamic programming and a discretized version of Shin and McKay's ACOTNI are applied.

INRODUCTION

In robotics two different controls are applied. For simple tasks like pick-and-place or spot welding the point-to-point (PTP) control is used. The programmer defines only the start and the end point of the robot motion (Fig. 1a). The actual path of the TCP is unknown and depends mainly on the specific robot dynamics and the feedback control on joint level. More demanding tasks like plasma welding, laser cutting or high speed operations in the presence of obstacles require an explicit knowledge of the complete path of the robot gripper and, in addition, a suitable velocity profile along that path (path-control; Fig. 1b). Simple path-control connects a set of control points by straight lines (path segments) resulting in a polygon with vertices (Fig. 1c). For accurate tracking of such a path the robot must stop at each vertex to move from one segment to the next. The successive procedure of acceleration and deceleration yields a very jerky robot motion and is time and energy consuming. Thus, for more satisfactory robot motion two problems must be solved: First, how to construct a smooth path with at least a second degree of continuity at the joints between adjacent curve segments, in order to be able to move from one path segment to the next without stopping the robot. Second, how to choose the velocity profile along the path in a somehow optimal way.

This paper presents the B-spline method for path planning for robots. The B-spline method has been developed in the past 30 years and is applied in Computer Aided Geometric Design (CAGD) and Computer Graphics to form smooth curves and surfaces (Newman and Sproull 1979), but it also fullfills the design specifications for curve generation for robots to a high degree. Furthermore, it is shown that B-spline functions are an excellent basis for some known trajectory planning algorithms. For generating optimal velocity profiles for B-spline curves Bellman's dynamic programming (Bellman 1957, Pun 1974, Shin and McKay 1986) and a discretized version of an algorithm proposed by Shin and McKay (1985) (ACOTNI) are introduced and compared.

In the following the problem related to construct a pure geometric path $\boldsymbol{P}(u)$ in cartesian or as well in configuration space is defined as path generation or path planning. The problem to find a suitable velocity profile $\dot{u} = \dot{u}(u)$; $u = u(t)$ for this

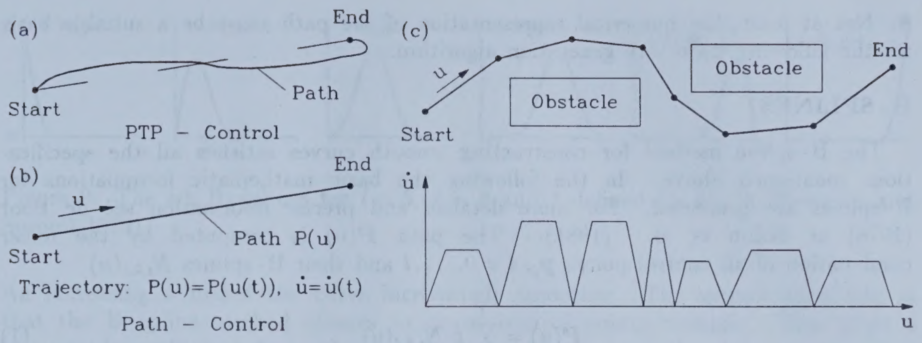


Figure 1: (a) PTP - control, (b) path - control, (c) polygon like path and appropriate velocity profile

path is defined as trajectory generation or trajectory planning because it makes a trajectory $\boldsymbol{P}(u(t))$ out of the path $\boldsymbol{P}(u)$; $u \neq u(t)$.

PATH GENERATION

Simple path control concepts do not smooth a given polygon like path by geometric modelling; they apply an uncontrolled smoothing: When the TCP comes into a certain range ϵ around a vertex $\boldsymbol{p}_i$ of the path, the direction of motion for the gripper is switched from $\boldsymbol{p}_i$ to the next vertex $\boldsymbol{p}_{i+1}$. The resulting deviation from the actual path is not known, i.e. is not controlled. Better path control concepts apply a variety of geometric shapes, most likely straight lines, circular and parabolic arc segments and cubic splines, to construct an allover smooth path (Bruhn and Ersü 1986). The problem of this approach lies in the difficult geometric design and the construction of the necessary degree of continuity at the connection points between the curve segments. A sophisticated and user - friendly path planning tool should meet the following specifications:

1: For convenience for any user, including automatic path planning, the rough structure of the path should be defined by the resulting polygon of consecutive control points. The desired smooth path must be predictively similar to this polygon and should not oscillate about the control points (variation - diminishing property).

2: The design of arbitrary paths requires multiple values, i.e. for one x - coordinate there could be many y - or z - coordinates and vice versa. This leads to a parametric vector representation of the curve $\boldsymbol{P}(u)$, where each component is expressed as a seperate univariate function of the path parameter u.

3: The shape of the path must be independent from the choosen coordinate system.

4: The designer should have a local control over the modelling of the path, i.e. the substitution of the position of one point in the chain of control points should alter the shape of the generated curve only in the area close to this point.

5: The user should have a broad variety of curve shapes to design the path.

6: The degree of continuity must be at least of second order at any point of the curve. For higher design flexibility and smoother curves it should be a freely eligible design parameter for the user.

7: Furthermore, the algorithm for computing the path should be of numerical ease and economic with memory.

8: Not at least, the numerical representation of the path must be a suitable basis for the following trajectory generation algorithm.

B-SPLINES

The B-spline method for constructing smooth curves satisfies all the specifications mentioned above. In the following the basic mathematic formulations for B-splines are presented. For more detailed and precise information see de Boor (1978) or Böhm et al. (1984). The path $\boldsymbol{P}(u)$ is computed by the linear combination of all control points $\boldsymbol{p}_i$, $i = 0, \ldots, l$ and their B-splines $N_{i,k,t}(u)$

$$\boldsymbol{P}(u) = \sum_{i=0}^{l} \boldsymbol{p}_i N_{i,k,t}(u) \quad . \tag{1}$$

The B-splines (some authers call them blending functions, i.e. Newman and Sproull 1979) $N_{i,k,t}(u)$ can be defined recursively:

$$N_{i,1,t}(u) = \begin{cases} 1 & \text{for} \quad t_i \le u < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,k,t}(u) = \frac{(u - t_i) N_{i,k-1,t}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u) N_{i+1,k-1,t}(u)}{t_{i+k} - t_{i+1}} \tag{2}$$

$$\text{for} \quad k \ge 2 \quad ; \quad i = 0, \ldots, l$$

with the knot values t_0 up to t_{l+k}

$$t_j = \begin{cases} 0 & \text{for} \quad j < k \\ j - k + 1 & \text{for} \quad k \le j \le l \\ j - k + 2 & \text{for} \quad j > l \end{cases} \quad ; \quad j = 0, \ldots, l + k \quad . \tag{3}$$

Since the denominator in Eq. (2) can become zero, the following holds by convention: $0/0 \equiv 0$. The definition range of the path parameter u follows from Eqs. (2) and (3):

$$0 \le u \le l - k + 2 \quad . \tag{4}$$

The parameter k controlls the continuity of the curve; the B-spline functions are of degree $k - 1$ and therefore $k - 2$-times continuously differentiable. Fig. 2 shows the six B-splines for a set of six control points ($l = 5$) for a quadratic B-spline function ($k = 3$) versus the curve parameter u ($0 \le u \le 4$). For $u = 0$ the control point $\boldsymbol{p}_0$ is weighted with 1 and all others with 0, i.e. the B-spline curve begins at the first control point; analogous is true at the last control point for $u = 4$. Through the B-splines the influence (weighting) of the control points on the path shape varies in dependence of the path parameter u. With exception of the start and end point ($u = 0$ and $u = l - k + 2$) exactly k (in Fig. 2 $k = 3$) control points are considered, since always only k B-splines are nonzero at a time. This way the B-spline method gives good local control in path planning.

For a fixed polygon B-splines offer two possibilities to vary the shape of the curve. The first is the parameter k, which controls the degree of continuity. The higher k is choosen, the more control points contribute to the lokal shape of the curve and this leads to larger deviations from the defining polygon. For $k = 1$ the B-spline is identical with the control points $\boldsymbol{p}_i$ and for $k = 2$ with the polygon.

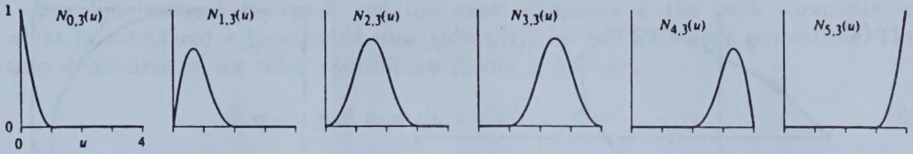


Figure 2: The six B-splines for $l = 5$, $k = 3$ and t defined by Eq. 3 (Newman and Sproull 1979)

An increasing k makes the curve increasingly smoother. The second possibility is that the B-spline method allowes to count control points multiple. This gives a respectively multiple influence of a certain control point on the path planning and the B-spline curve passes this point closer; in the extrem vertices can be created this way.

The characteristics of the B-spline method make it a powerfull path planning tool in robotics. In contrast to other approaches (Bruhn and Ersü 1986, Rojek 1987), in which each single vertex is smoothed with circular arcs or splines, thus only local smoothing is done, B-splines make global smoothing possible, i.e. the path is defined from the beginning to the end by only one single B-spline function and does not consist of many segments. This great advantage makes the path planning process much more comfortable for the user and faster in computing.

In our department B-splines are succesfully applied for constructing smooth paths for industrial robots. In one application B-spline functions serve as a basis for optimal trajectory generation (Schmidt 1990). In a second application B-splines are used to smooth the vertices of a polygon, which is the result of a task-level programmed algorithm to automatically search collision-free motions for robots in the presence of obstacles (Fink and Wend 1989). Note that the programming on task-level requires an automatic smoothing; adjustments or corrections by the user are not possible. From our experiences the following rules of thumb can be formulated for path planning for robots with B-splines:

1: For best efficiency of the method, the path parameter u should have a fixed discretization $1/\Delta u = g,\ g \in N$. The path $\boldsymbol{P}(u)$ is then represented by $u_{max}/\Delta u + 1 = (l - k + 2)g + 1$ points.

2: The degree of continuity of the curve should be 3 or 5 ($k = 4$ or 6).

3: The control points should possibly lie distributed equidistantly.

4: For the smoothing of vertices the corresponding control points are counted double.

5: Arched shapes, which must be tracked very accurately are defined by their sourrounding polygon. Start and end point of the arc count double; the other control points count single. Further control points are put at the tangent points between the polygon and the arc.

6: Start and end point of the curve need to be counted only single.

A two-dimensional cubic ($k = 4$) B-spline curve and its defining polygon are given in Fig. 3a for a case where all control points count single ($l = 13$). Fig. 3b shows the approximation of a semicircle with B-spline functions.

TRAJECTORY GENERATION

From the proposed techniques for solving the minimum time problem for robot motion using the manipulator full dynamics, only some formulate the optimization

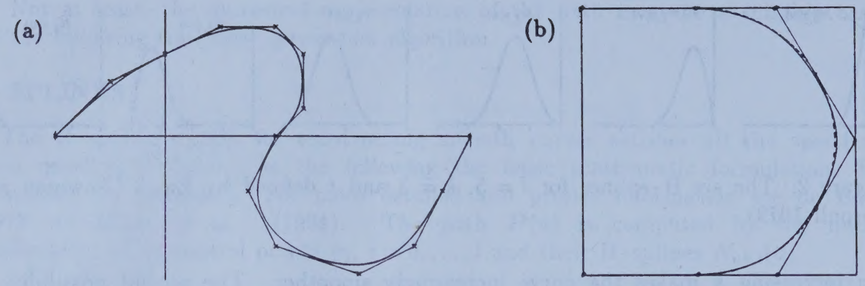


Figure 3: (a) Cubic B - spline curve with defining polygon and (b) circular arc $(\cdots)$ approximation with B - spline functions $(-)$

problem for a given geometric path which must be tracked (Vukobratović and Kirćanski 1982, Hollerbach 1984, Bobrow et al. 1985, Shin and McKay 1985, Johanni and Pfeiffer 1987). The solution algorithms introduced in this paper to find optimal velocity profiles along curves generated with the B - spline method are based on the very close approaches from Shin and McKay (1985 and 1986) and Johanni and Pfeiffer (1987).

OPTIMIZATION PROBLEM

Given are the parametric path $\boldsymbol{P}(u)$ in configuration space[1]

$$\boldsymbol{P}(u) = \begin{pmatrix} q_1(u) \\ \vdots \\ q_n(u) \end{pmatrix} \quad ; \quad 0 \leq u \leq u_{max} \tag{5}$$

and the full robot dynamics

$$\tau_i = \sum_{j=1}^{n} m_{ij}(\boldsymbol{q})\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} m^{i,jk}\dot{q}_j\dot{q}_k + d_i^*\dot{q}_i + g_i(\boldsymbol{q}) \quad ; \quad i = 1,\dots,n \tag{6}$$

where	τ_i	are the generalized joint forces
	$m_{ij}(\boldsymbol{q})$	are the elements of the $n \times n$ kinetic energy matrix
	$\boldsymbol{q}$	is the vector of the n generalized joint coordinates q_i
	$m^{i,jk}$	$= \frac{1}{2}\left(\frac{\partial m_{ij}(\boldsymbol{q})}{\partial q_k} + \frac{\partial m_{ik}(\boldsymbol{q})}{\partial q_j} - \frac{\partial m_{jk}(\boldsymbol{q})}{\partial q_i}\right)$ is the Christoffel symbol
	d_i^*	are the coefficients for viscous friction
	$g_i(\boldsymbol{q})$	are the gravitational forces.

Furthermore, the following constraints must be considered

$$\tau_{i,min} \leq \tau_i \leq \tau_{i,max} \quad ; \quad \dot{q}_{i,min} \leq \dot{q}_i \leq \dot{q}_{i,max} \quad ; \quad \dot{u} \leq \dot{u}_{max} \tag{7}$$

which take limited torques, joint velocities and absolute path velocity into acount. The minimum time path planning problem is to minimize the overall traversal time T to move from the start to the end point of $\boldsymbol{P}(u)$ subject to Eqs. (5) to (7).

---

[1] If the path planning takes place in cartesian coordinates the path can easily be transformed pointwise into configuration space.

The link between the path and the robot dynamics is the path parameter u, which is considered a function of time ($u = u(t)$) for the trajectory generation. The time derivatives of the robot coordinates $\dot{q}_i$ and $\ddot{q}_i$ become

$$\dot{q}_i = q'_i \dot{u} \text{ and } \ddot{q}_i = q'_i \ddot{u} + q''_i \dot{u}^2 \quad ; \quad i = 1, \ldots, n \tag{8}$$

where q' denotes dq/du and $q'' = d^2q/du^2$. Putting Eq. (8) into Eq. (6) yields

$$\tau_i = a_i \ddot{u} + b_i \dot{u}^2 + d_i \dot{u} + g_i \tag{9}$$

with $a_i = \sum_{j=1}^{n} m_{ij}(\boldsymbol{q}) q'_j \quad ; \quad b_i = \sum_{j=1}^{n} m_{ij}(\boldsymbol{q}) q''_j + \sum_{j=1}^{n} \sum_{k=1}^{n} m^{i,jk} q'_j q'_k \quad ; \quad d_i = d^*_i q'_i \quad .$

The path parameter u transforms the multi-dimensional optimization problem described by Eqs. (5) to (7) to a one-dimensional problem given by Eqs. (5), (7) and (9) and makes possible efficient solution algorithms in the phase plane $u - \dot{u}$. In addition, with Eq. (8) the total time T can be written as

$$T = \int_0^{t_E} dt = \int_{u_{min}}^{u_{max}} \frac{dt}{du} du = \int_{u_{min}}^{u_{max}} \frac{1}{\dot{u}} du \quad , \tag{10}$$

i.e. minimizing T is equal to maximizing the path velocity[2] $\dot{u}$.

BOUNDARY CURVE

In the two approaches which are introduced here, the first step to solve the optimization problem is the same and that is to find the maximum velocity $\dot{u}$ for each point u on the path. This gives a boundary curve $\dot{u}_{BC}(u)$ in the $u - \dot{u}$ plane. Putting the torque limits (7) in Eq. (9) and rearanging gives

$$\ddot{u}_{i,ex}(u, \dot{u}) = -\frac{b_i}{a_i}\dot{u}^2 - \frac{d_i}{a_i}\dot{u} + \frac{\tau_{i,ex} - g_i}{a_i} \quad , \tag{11}$$

where the acceleration boundaries $\ddot{u}_{i,ex}$ are expressed as a quadratic funktion of the velocity[3] $\dot{u}$ for a fixed u. By convention, $\ddot{u}_{i,ex}$ is considered $\pm\infty$ for $a_i = 0$. This regards the fact that in the case of $\sum_{j=1}^{n} m_{ij}(\boldsymbol{q}) q'_j = 0$ no motion takes place in the direction of joint i and there are no coupling effects on this joint. The $2i$ parabolic curves $\ddot{u}_{i,UB}(u, \dot{u})$ and $\ddot{u}_{i,LB}(u, \dot{u})$ define in the $\dot{u} - \ddot{u}$ plane an area of admissable accelerations and velocities for a specific u (Fig. 4a). The maximum possible velocity $\dot{u}_{BC}$ is bounded by either the dynamic equations (11) or the velocity constraints $\dot{u} \leq \dot{u}_{max}$ and $\dot{u} \leq \dot{q}_{i,ex}/q'_i$, $\forall i$. In the phase plane, $\dot{u}_{BC}(u)$ represents an upper velocity boundary (boundary curve) for all possible velocity profiles along the path (Fig. 4b).

DYNAMIC PROGRAMMING

A wellknown method to solve one-dimensional optimization problems is Bellman's dynamic programming (Bellman 1957, Pun 1974). It is proposed in Shin and McKay (1986) to solve the optimal trajectory planning problem for arbitrary

---

[2] Note that because u is an artificial path parameter $\dot{u}$ is also an artificial velocity without physical meaning.

[3] It is assumed that the torque limits $\tau_{i,ex}$ are at most quadratic functions of the joint velocities q_j, $j = 1, \ldots, n$.

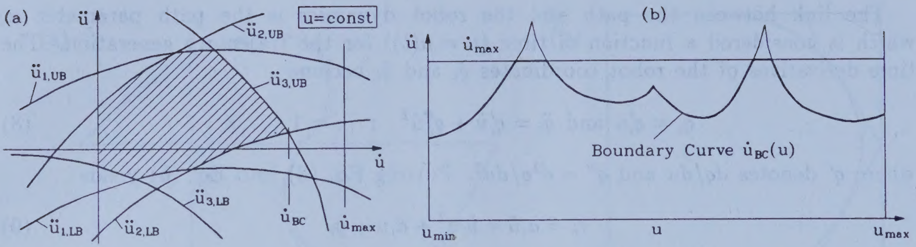

(a)
$\ddot{u}$
u=const
$\ddot{u}_{2,UB}$
$\ddot{u}_{1,UB}$
$\ddot{u}_{3,UB}$
$\dot{u}$
$\ddot{u}_{3,LB}$
$\ddot{u}_{1,LB}$
$\ddot{u}_{2,LB}$
$\dot{u}_{BC}$
$\dot{u}_{max}$
(b)
$\dot{u}$
$\dot{u}_{max}$
Boundary Curve $\dot{u}_{BC}(u)$
u_{min}
u
u_{max}

Figure 4: (a) Area of admissable accelerations and velocities, (b) boundary curve

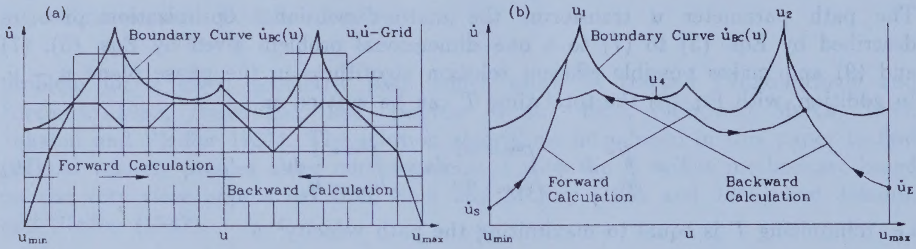

(a)
$\dot{u}$
Boundary Curve $\dot{u}_{BC}(u)$
u,$\dot{u}$-Grid
Forward Calculation
Backward Calculation
u_{min}
u
u_{max}
(b)
$\dot{u}$
u_1
u_2
Boundary Curve $\dot{u}_{BC}(u)$
u_4
Forward Calculation
Backward Calculation
$\dot{u}_S$
$\dot{u}_E$
u_{min}
u
u_{max}

Figure 5: Phase plane for (a) dynamic programming and (b) ACOTNI

cost functions. For calculations the phase plane has to be discretized in u and $\dot{u}$ giving a grid of position-velocity points (Fig. 5a). The boundary curve $\dot{u}_{BC}$ from step 1 provides information to find a propper velocity discretization and to speed up the algorithm. Then, starting from the end, the optimal trajectory for each possible velocity for a specific position u to the final position and velocity must be computed and stored. The algorithm moves stepwise from u_{max} to u_{min}. Is u_{min} reached, the optimal velocity profile can be traced back from the stored data. The advantages of the dynamic programming are the arbitrary optimization criterion and the relative easy programming of the algorithm. However, there are some strong drawbacks. First, the velocity $\dot{u}$ must be discretized and good solutions require a small discretization. Second, the need to store all optimal trajectories for all possible velocities leads to a huge memory consumption. Finally, the calculations require a relative long time.

ACOTNI

For the time optimal solution Shin and McKay (1985) proposed a fast and efficient algorithm for constructing optimal trajectories (ACOTNI). This algorithm does not require a discretization of neither u nor $\dot{u}$ if the path and the derivatives q_i' and q_i'' are given analytically, but requires tedious symbolic derivations for each different path or even path segment. A far more general algorithm can be programmed when the path is given in a discrete way (discrete u) and when *difference* quotients are used for q_i' and q_i'' instead of *differential* quotients. Since the B-spline method works very efficient with a discrete path parameter, too, it is an excellent basis for the discrete ACOTNI and for the dynamic programming approach.

The search algorithm ACOTNI works as follows (see Fig. 5b):

Step 1: Starting from $(u_{min}, \dot{u}_S)$ a trajectory with maximum possible acceleration $\ddot{u}_{max}(u, \dot{u}) = \min[\ddot{u}_{i,UB}(u, \dot{u})\ ,\ i = 1, \ldots, n]$ is constructed forwards. The accelerating curve ends either at u_{max} or when it leaves the admissible region, i.e. when $\ddot{u}_{min}(u, \dot{u}_{BC}) < \ddot{u}_{BC}$ $(u = u_1$ in Fig. 5b).

Step 2: Starting from $(u_{max}, \dot{u}_E)$ a trajectory with maximum possible deceleration $\ddot{u}_{min}(u, \dot{u}) = \max[\ddot{u}_{i,LB}(u, \dot{u})\ ,\ i = 1, \ldots, n]$ is constructed backwards. The decelerating curve ends either when it intersects the accelerating curve or when it leaves the admissible region, i.e. when $\ddot{u}_{max}(u, \dot{u}_{BC}) < \ddot{u}_{BC}$ $(u = u_2$ in Fig. 5b).

Step 3: In case of intersection the optimal trajectory is found. It starts in $(u_{min}, \dot{u}_S)$, goes along the accelerating curve until the point of intersection and then along the decelerating curve to the end point $(u_{max}, \dot{u}_E)$.

Step 4: In case of no intersection, starting at the point where the accelerating curve leaves the admissable region $(u = u_1)$, a search is done along $\dot{u}_{BC}$ for the first point, from which trajectories can be computed into the admissible region, i.e. where $\ddot{u}_{min}(u, \dot{u}_{BC}) \leq \ddot{u}_{BC}$ $(u = u_4$ in Fig.5b).

Step 5: Similar to step 2 but starting from u_4 a decelerating curve is constructed backwards until it intersects an accelerating curve.

Step 6: Similar to step 1 but starting from u_4 an accelerating curve is constructed forwards until it intersects the final decelerating curve or leaves the admissible region. In case of intersection, the optimal trajectory is found, consisting of the successive intersecting accelerating and decelerating curves. Otherwise the procedure starts again from step 4.

Compared to the dynamic programming approach this algorithm has the advantages of continuous velocity $\dot{u}$, small memory requirements and much faster computing time (factor 10 to 1000). At the moment, the drawback is that only the time minimum problem can be solved with ACOTNI.

APPLICATION EXAMPLE

Fig. 6 shows the results of constructing a smooth path with B-splines and the corresponding time optimal trajectory for a SCARA robot (joints 1 and 2). It can be seen from the switching of the accelerations (one joint acceleration is allways saturated) that the proposed algorithms find the correct solution for the complicated path, but ACOTNI works much better than dynamic programming. On a HP 9000/835 computer the computation times for path and trajectory are 1 s for the dynamic programming and 0.13 s for ACOTNI.

CONCLUSIONS

In this paper it is shown that efficient and convenient path and trajectory generation are two interdependent problems which should be solved in regard to each other. The B-spline method is a powerfull tool for planning arbitrary, smooth, geometric paths with a freely eligible degree of continuity. Furthermore, its mathematical representation serves as a good basis for optimal trajectory planning. For the trajectory generation problem Bellman's dynamic programming and Shin and McKay's ACOTNI are applied. By discretising the path, a general and fast algorithm can be realized using a discrete ACOTNI. This combination makes it possible to generate optimal trajectories for robots for arbitrary geometric curves constructed in cartesian or configuration space. However, it seems that the dynamic programming approach is too time and memory consuming for practical applications

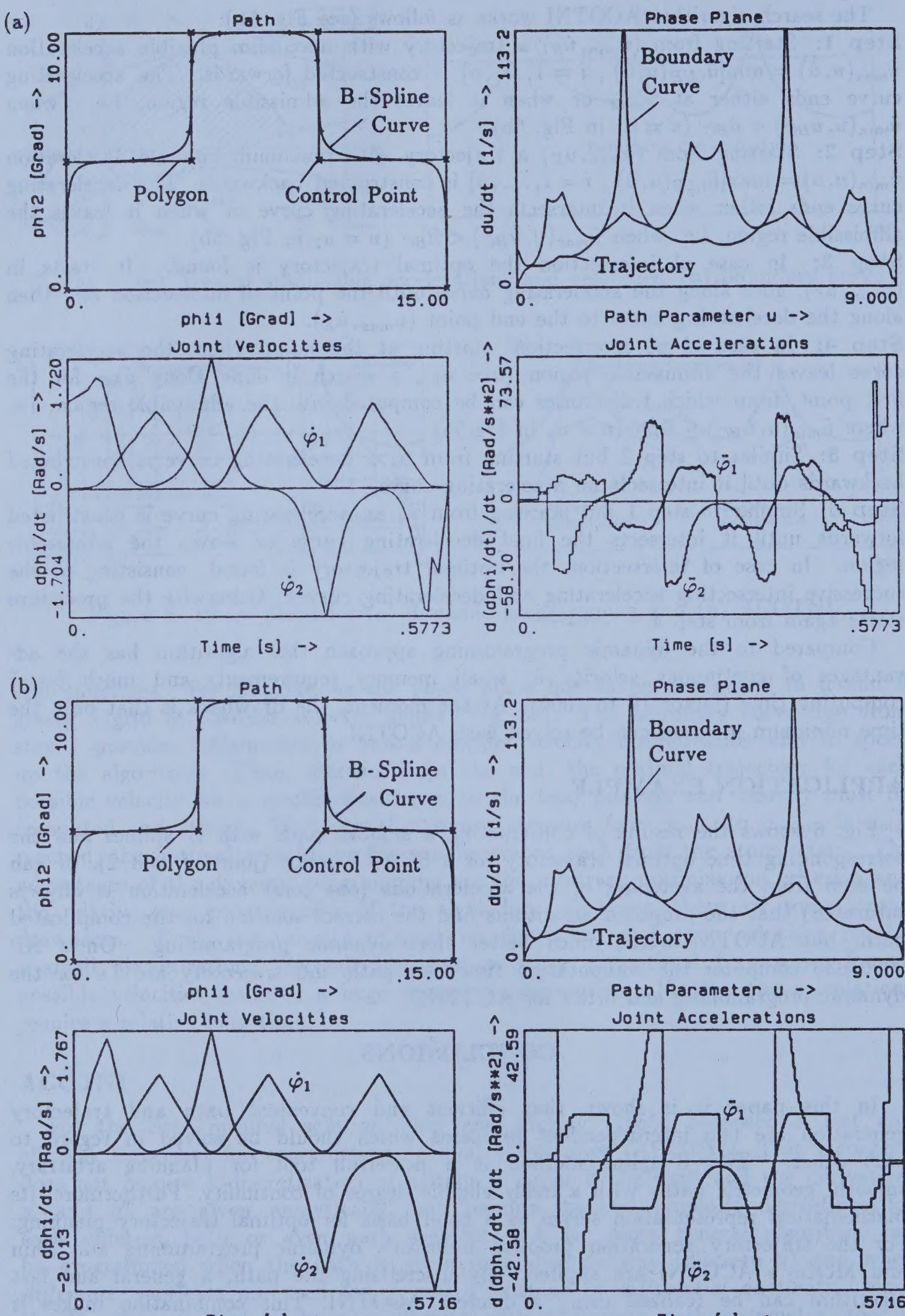


Figure 6: Simulation results for (a) dynamic programming and (b) ACOTNI

while the resulting unlimited jerks of the time minimum solution efficiently found with ACOTNI hinder practical use. Thus, our research concentrates on procedures to smooth the optimal trajectory in the phase plane to limit the jerks.

REFERENCES

Bellman, R. 1957. *Dynamic Programming.* Princetown (New Jersey): Princeton University Press.

Bobrow, J. E., S. Dubowsky and J. S. Gibson. 1985. Time-Optimal Control of Robotic Manipulators Along Specified Paths. *Int. J. Robotics Research* 4. 3–17.

Böhm, W., G. Farin and J. Kahmann. 1984. A Survey of Curve and Surface Methods in CAGD. *Computer Aided Geometric Design* 1. 1–60.

de Boor, C. 1978. *A Practical Guide to Splines.* New York: Springer-Verlag.

Bruhn, H. and **E. Ersü.** 1986. Grundlagen der analytischen Bewegungsplanung für Roboter. *VDI Bericht* Nr. 598.

Fink, B. and **H.-D. Wend.** 1989. A Fast Path-Planning Algorithm for Industrial Robots. Proceedings *IEEE International Conference on Control and Applications.* Jerusalem (Israel). WA-3-2.

Hollerbach, J. M. 1984. Dynamic Scaling of Manipulator Trajectories. *Journal of Dynamic Systems, Measurement and Control* 106.

Johanni, R. and **F. Pfeiffer.** 1987. Optimale Bahnplanung für Industrieroboter. *Robotersysteme* 3. 29–36.

Newman W. M. and **R. F. Sproull.** 1979. *Principles of Interactive Computer Graphics.* 2nd edition. New York: McGraw-Hill Book Company.

Pun, L. 1969. *Introduction to Optimization Practice.* New York: John Wiley & Sons, Inc.

Rojek, P. 1987. *Bahnführung eines Industrieroboters mit Multiprozessorsystem.* Dissertation Universität Braunschweig.

Schmidt, F. 1990. *Trajektorienplanung für Industrieroboter.* Diplomarbeit MSRT. Universität Duisburg.

Shin, K. G. and **N. D. McKay.** 1985. Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints. *IEEE Trans. on Automatic Control* AC-30. 531–541.

Shin, K. G. and **N. D. McKay.** 1986. A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators. *IEEE Trans. on Automatic Control* 31. 491–500.

Vukobratović, M. and **M. Kirćanski.** 1982. A Method for Optimal Synthesis of Manipulation Robot Trajectories. *Journal of Dynamic Systems, Measurement and Control.* 104. 188–193.

Algorithms and Bounds for Shortest Paths and Diameter in Faulty Hypercubes*

Sing-Ban Tien and C.S. Raghavendra
Department of Electrical Engineering – Systems
University of Southern California
Los Angeles, Ca 90089- 0781
raghu@pollux.usc.edu

Abstract

In this paper, efficient algorithms are developed for finding near shortest paths in hypercubes containing faulty nodes and/or links. In an n–dimensional hypercube Q_n with a set $|F| < n$ faulty components – nodes and/or links, an efficient algorithm of complexity $O(n \log^2 |F|)$ is given to find a near shortest path between any two nodes S and D. The path length is at most their Hamming distance plus 2. When the fault set $|F| < 2n - 2$, assuming S and D are not isolated, it is shown that there exists a path with length at most their Hamming distance plus 4. An algorithm is given to find such a path in $O(n^2 \log |F|)$ steps. Further, we obtain a bound for the diameter of the faulty hypercube $Q_n - F$, when $|F| < 2n - 2$ and $Q_n - F$ is connected, as $n + 2$. This improves the previously known bound of $n+6$ by Esfahanian [6]. Worst case scenarios are constructed to show that these bounds for shortest paths and diameter are tight.

Index terms: multiprocessor, hypercube, fault tolerance, routing, shortest path, diameter.

1 Introduction

Multiprocessor systems with large number of processors are becoming feasible due to advances in technology and low cost of hardware. Due to the rich connectivity and its regular structure, hypercube [16] has become a popular interconnection network topology. Hypercube multiprocessors are now available commercially through several companies such as Intel, Amtek, Ncube, Floating Point Systems, etc. Hypercube multiprocessors have been the focus of extensive research in recent years. Various algorithms have been proposed for routing, broadcasting and related problems [2, 10, 13, 19] as well as for applications [4, 9, 12].

Another important issue of hypercube multiprocessors is fault tolerance due to the following reasons: the probability that some processor becomes faulty among such a large number of processors will be high, and a single node or link failure can destroy the regularity and thus may disrupt the entire task execution. There is a need to provide fault tolerance capabilities for hypercube multiprocessors. Many hardware and software techniques have been proposed such as [1, 11, 14, 18]. The research on this subject motivates the need to understand how the basic properties of hypercubes are affected by faults. Such results can help the development of better fault tolerance techniques. In this paper, we focus on how the length of shortest paths between arbitrary two nodes and the diameter of hypercubes are affected by faults.

Since the connectivity of an n-dimensional hypercube is n, when the number of faults is less than n, the hypercube remains connected. It has been shown that there are n node disjoint paths between any two nodes and when number of faults is less than n the length of the shortest path between the two nodes is increased by at most 2 [15]. Diameter of hypercube is also known to be increased by at most 1 when number of faults is less than n [8]. When number of faults is more than the connectivity, n, the hypercube can become disconnected. Therefore, the shortest path problem and diameter problem may not be a meaningful measure of fault tolerance. However, connectivity is a worst-case measure in the sense that n node failure can disconnect a hypercube only if they are neighbors of a particular node. In practice, this is not likely to happen. There is a need to study the issues in non-worst-case situations.

In [6], the author proposed the concept of a *forbidden set.* Based on this concept, he showed that an n-dimensional hypercube remains connected when there are less than $2n - 2$ faults, provided that the neighbors of any node do not become faulty at the same time — the forbidden set. He also showed that under this situation, the diameter is bounded by $n + 6$. In this paper, we follow this line of investigation and probe further.

In this paper, we consider both link and node failures. There are usually two fault models for node. One model assumes that the computational function of the node is lost while the communication function remains intact. Therefore, nonfault-tolerant communication schemes are applicable directly. The other model assumes that the com-

---

*This research is supported by the NSF Grant MIP8452003

munication function is lost; therefore it renders the computational function useless regardless it is faulty or not. The later model is more general and interesting so it is assumed in this paper.

When the number of faulty components, faulty links and/or nodes, is less than n, the existence of a path of length at most their Hamming distance plus 2 between any two nodes is known [15]. We give an efficient algorithm to find it, with complexity $O(n \log^2 |F|)$, where n is the number of dimensions and $|F|$ is the size of faulty components. In contrast, a naive algorithm takes $O(n^2 \log |F|)$ unit times.

When the number of faulty components is less than $(2n-2)$, we show that between any two nonisolated nodes there exists a path of length at most their Hamming distance plus 4. An algorithm of complexity $O(n^2 \log |F|)$ is given to find such a path. Worst case scenarios are constructed to show this bound for shortest path is tight.

This result shows that the diameter of $Q_n - F$ when connected is at most $n+4$. However, when the distance between the two nodes is n or $(n-1)$, we show that a path of length n or at most $(n+1)$, respectively, can be found. Consequently, the longest path with length $n+2$ occurs when two nodes are distance $n-2$ apart. Hence, the diameter of $Q_n - F$ is at most $n+2$ when $|F| < 2n-2$ and $Q_n - F$ is connected. Worst case scenarios are constructed to show this bound for diameter is tight. Best known previous result is $n+6$ [6] for the diameter of connected $Q_n - F$, $|F| < 2n-2$.

This paper is organized as follows. In section 2, some notations and definitions are given. Section 3 presents an efficient algorithm for finding a near shortest path when the number of faults is less than n. In section 4, we present bounds on shortest paths and diameter of connected faulty hypercubes when the number of faulty components is greater than $n-1$ but less than $2n-2$. We also show that these bounds are tight. An algorithm is also given to find a path with length within the bound. Finally, some conclusions are given in section 5.

2 Preliminaries

An n-dimensional hypercube consists of 2^n nodes where each node is labeled with a binary number. There is a link between two nodes if their labels differ in exactly one bit. The label of a node is usually represented as $(b_n, \ldots, b_1)$, $b_i = 0, 1$, for $1 \leq i \leq n$. Two nodes have distance k if their labels differ in k bits. This distance is referred to as Hamming distance. In the rest of the paper, n will always denote the number of dimension of a hypercube. Basic results concerning hypercube can be found in [7, 15].

We also assume that most logic operations such as **and**, **or**, and **exclusive-or** on n-bit labels can be performed in unit time by storing a label in a word. This is possible since most practical machines have word length 32 and the dimension of pratical hypercube multiprocessors is no more than 16.

In this paper, we consider finding a path between a source node S and a destination node D in a Q_n containing faulty components. In the following, we simply say fault instead of faulty component. S and D are assumed to be fault-free. For a node $X = (x_n \ldots x_1)$, its neighbor across dimension i, $(x_n \ldots \bar{x}_i \ldots x_1)$, is denoted as $\bar{X}_i$. To generalize this notation a little further, we let $\bar{X}_{i,j} = (x_n \ldots \bar{x}_i \ldots \bar{x}_j \ldots x_1)$ and similarly for complementing 3 bits. The Hamming distance between two nodes $X = (x_n \ldots x_1)$ and $Y = (y_n \ldots y_1)$ is denoted as $H(X,Y)$. Let $I(X,Y) = \{i | x_i \neq y_i\}$, $J(X,Y) = \{i | x_i = y_i\}$ and $Q_m(S,D)$ denote the m-dimensional subcube(m-subcube) containing S and D. When $m = H(S,D)$, $Q_m(S,D)$ is the minimum subcube containing S and D. A path between S and D, denoted as $P(S,D)$, is a minimal path if its length, denoted as $|P(S,D)|$, is equal to $H(S,D)$. $P(S,D)$ is represented as a sequence of dimensions that the path traverses starting from S and ending at D. An m-subcube is represented as a ternary string of $\{0,1,*\}$. For example, the 2-subcube $\{000, 001, 010, 011\}$ is represented as $0**$. We denote $\bar{X}_i$ is accessible to X by $\bar{X}_i \leftarrow X$ (or $X \rightarrow \bar{X}_i$) if both $\bar{X}_i$ and the link between X and $\bar{X}_i$ are fault-free, otherwise by $\bar{X}_i \not\leftarrow X$(or $X \not\rightarrow \bar{X}_i$). A node is isolated if all its neighbors are inaccessible. Throughout this paper we let F denote the set of faulty components.

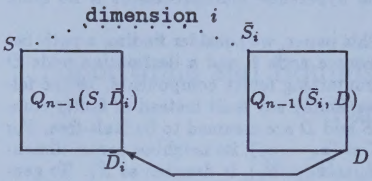
3 Number of Faults Less Than n

We first consider the case that $|F|$ is less than n. In [15], it is shown that there are n node disjoint paths for two nodes X and Y in a fault-free n-dimensional hypercube. Among them, $d = H(X,Y)$ paths are minimal paths with length equal to $H(X,Y)$ and the rest are of length $H(X,Y)+2$. Therefore, when $|F| < n$, there exists a path of length at most $H(X,Y)+2$. The question is how fast we can find such a path. Every path can have $O(n)$ nodes in the worst case. A naive algorithm, which checks every node of the n node-disjoint paths($O(n^2)$ nodes) against the sorted list of faults, takes $O(n^2 \log |F|)$ time units. In this section, we present an efficient algorithm which can find a path of length at most $H(X,Y)+2$ in $O(n \log^2 |F|)$ time units.

3.1 The Distance between S and D is n.

We reprove the following known result in an algorithmic way and then construct an algorithm out of the proof.

Lemma 3.1 *If $H(S,D) = n$ and $|F| < n$, then there exists a path $P(S,D)$ with length $|P(S,D)| = n$.*



$$F \bigcap Q_{n-1}(S, \bar{D}_i) \leq F \bigcap Q_{n-1}(\bar{S}_i, D)$$

Figure 1: Reduction of routing to a smaller subcube with at most half of $|F|$ faults.

Proof:
Since there are less than n faults, there must be a dimension i such that $S \rightarrow \bar{S}_i$ and $\bar{D}_i \leftarrow D$. As depicted in Fig. 1, we can split the n-cube along dimension i into two $(n-1)$-subcubes. One of the two $(n-1)$-subcubes has at most the same number of faults as the other and therefore contains at most half of the faults. Routing from S to D can be reduced to routing in this $(n-1)$-subcube. After at most $\lceil log|F| \rceil$ times of reduction, a fault-free subcube can be found and routing in that subcube is straightforward. A formal proof is given below.

Let

$$D_s = \{i | S \not\rightarrow \bar{S}_i, 1 \leq i \leq n\}$$

and

$$D_d = \{i | \bar{D}_i \not\leftarrow D, 1 \leq i \leq n\}$$

We have $|D_s| + |D_d| \leq |F| < n$ and this implies $|\{1 \ldots n\} - D_s - D_d| > 0$. Therefore, there exists some i such that $S \rightarrow \bar{S}_i$ and $\bar{D}_i \leftarrow D$. Without loss of generality, we can assume

$$|F \bigcap Q_{n-1}(S, \bar{D}_i)| \leq |F \bigcap Q_{n-1}(\bar{S}_i, D)|$$

Now we can reduce our problem to finding $P(S, \bar{D}_i)$ in $Q_{n-1}(S, \bar{D}_i)$ with $|F \bigcap Q_{n-1}(S, \bar{D}_i)| \leq \lfloor n/2 \rfloor < n-1$ faults, as shown in Fig. 1.

$$P(S, D) = P(S, \bar{D}_i) \circ i,$$

where $\circ$ denotes concatenation. Therefore, $P(S, D)$ can be recursively routed. After at most $\lceil \log |F| \rceil$ times of reduction, a subcube with no fault is found and routing in the fault-free subcube is straightforward. It is not hard to see that $|P(S, D)| = H(S, D) = n$. □

We now construct an algorithm out of the proof of Lemma 3.1 to find a minimal path between two nodes with distance n.

Algorithm n-Route

```
Sort F;
n-Route(S,D); /* n-Route(S,D) is a recursive
    procedure defined below.*/
Procedure n-Route(S,D);
/*Return a sequence of dimensions*/.
begin
    $m = H(S, D)$;
    If $F \bigcap Q_m(S, D) = \emptyset$ then;
        return $I(S, D)$;
        exit;
    else begin
        $i = 0$;
        Repeat /*Search for a dimension $i$ where
        for both $S$ and $D$ their neighbors across
        dimension $i$ are accessible to them.*/
        i=i+1;
        until $S \rightarrow \bar{S}_i$ and $\bar{D}_i \leftarrow D$;
        Split $Q_m(S, D)$ along dimension $i$ into
        $Q_{m-1}(S, \bar{D}_i)$ and $Q_{m-1}(\bar{S}_i, D)$;
        Compute $Q_{m-1}(S, \bar{D}_i) \bigcap F$
        and $Q_{m-1}(\bar{S}_i, D) \bigcap F$;
        If $|Q_{m-1}(S, \bar{D}_i) \bigcap F| \leq |Q_{m-1}(\bar{S}_i, D) \bigcap F|$
        then $P(S, D) =$ n-Route$(S, \bar{D}_i) \circ i$
        else $P(S, D) = i \circ$ n-Route$(\bar{S}_i, D)$;
        return $P(S, D)$;
    end;
end;
```

First, we sort F. There are two sorted lists: one for faulty nodes and the other for faulty links. It takes $O(|F| \log |F|)$ steps by standard techniques. Computing $F \bigcap Q_m(S, D)$ can be done in $O(|F|)$ by going through the fault set. Computing the sequence of dimensions where they differ takes $O(n)$.(Exclusive-or S and D, then shift out bits one by one and a 1-bit indicates S and D differ.) If $F \bigcap Q_m(S, D) \neq \emptyset$, it takes $O(n \log |F|)$ to find a dimension i such that both $S \rightarrow \bar{S}_i$ and $\bar{D}_i \leftarrow D$. For each i, 2 binary searches, one for the connecting i-th dimensional link and the other for the neighbor across dimension i, need be performed for $S(D)$ in sorted F to ensure both the link between $S(D)$ and $\bar{S}_i(\bar{D}_i)$ and node $\bar{S}_i(\bar{D}_i)$ are fault-free. After this, n-Route is called recursively for a smaller dimension cube and at most half of the size of the fault set. The complexity of a single run of the procedure is $O(n \log |F|)$. Since the procedure can at most be invoked $\lceil \log |F| \rceil$ times, complexity of Algorithm n-Route is $O(n \log^2 |F|)$.

A slight modification of n-Route can make it tolerate a bigger fault set $|F| < n + \lceil n/2 \rceil - 1$, with the assumption that S and D are not isolated. If there is a common i such that $S \rightarrow \bar{S}_i$ and $\bar{D}_i \leftarrow D$, the fault set can be reduced by half and recursively route through in the same way. If there is no such i , then it can be seen that either S or D must have at least $\lceil n/2 \rceil$ inaccessible neighbors. Say S does, with the assumption that neither S nor D is isolated, pick an i such that $S \rightarrow \bar{S}_i$ and split the cube. Then

$$|Q_{n-1}(\bar{S}_i, D) \bigcap F| < n + \lceil n/2 \rceil - 1 - \lceil n/2 \rceil = n - 1$$

Now **Algorithm n-Route** can be called with source $\bar{S}_i$, destination D, and fault set $Q_{n-1}(\bar{S}_i, D) \bigcap F$.

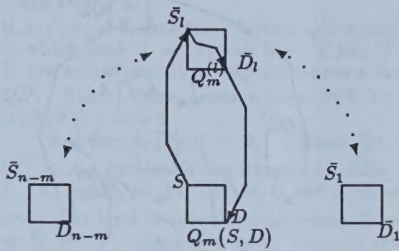


Figure 2: One of the adjacent m-subcube must be fault-free inside and among the links between itself and $Q_m(S,D)$, and a path can go through it.

3.2 The Distance between S and D is Less Than n.

Let $H(S,D) = m < n$. If $|Q_m(S,D) \bigcap F| < m + \lceil m/2 \rceil - 1$ and both S and D are not isolated in $Q_m(S,D)$, algorithm n-Route can be used to find a minimal path. Otherwise, as indicated in [15], there are $(n-m)$ m-subcubes adjacent to $Q_m(S,D)$ and the path needs to go through one adjacent m-subcube. Since there are at least m faults in $Q_m(S,D)$, at most $n-1-m$ fault are located elsewhere. Therefore, out of $(n-m)$ adjacent m-subcubes, one $Q_m^{(l)}$ must be fault-free and have all links between $Q_m^{(l)}$ and $Q_m(S,D)$ fault-free, where $Q_m^{(l)}$ is the m-subcube adjacent to $Q_m(S,D)$ across dimension l. We denote this by $Q_m^{(l)} \leftarrow Q_m(S,D)$ and otherwise by $Q_m^{(l)} \not\leftarrow Q_m(S,D)$. So $P(S,D) = l \circ P(\bar{S}_l, \bar{D}_l) \circ l$ as shown in Fig. 2 and $|P(S,D)| = m+2$.

The general algorithm for routing from S to D in Q_n with $|F| < n$ first checks to see if **n-Route** can be used to find a minimal path. If **n-Route** can not be applied, it then searches for a $Q_m^{(l)} \leftarrow Q_m(S,D)$. The complete algorithm, denoted **Algorithm 1** is given below.

Algorithm 1

```
Sort F; /*one list for faulty nodes and the other for faulty links */
Compute $I(S,D)$ and $J(S,D)$, and $H(S,D)$;
Determine if $S$ or $D$ is isolated in $Q_m(S,D)$;
$m = H(S,D)$;
If $H(S,D) = n$, or ($|F \bigcap Q_m(S,D)| < m + \lceil m/2 \rceil - 1$ and both $S$ and $D$ are not isolated in $Q_m(S,D)$)
then run Algorithm n-Route and exit;
/* Otherwise search for an adjacent m-subcube $Q_m^{(l)} \leftarrow Q_m(S,D)$. */
For each $X \in F$,
    (i) $X = (x_n \ldots x_1)$ a node, set $x_i = 0$,
        for all $i \in I(S,D)$;
    (ii) $X = (x_n \ldots x_{i+1} 1 x_{i-1} \ldots x_1, i)$ a link,
        set $x_i = 0$ for all $i \in I(S,D)$.
        if $i \in I(S,D)$, this implies that $X$ is a faulty link inside an adjacent m-subcube or
        $Q_m(S,D)$. Since in this case $X$ serves the same purpose as a faulty node, $X$ is treated
        and represented as a faulty node by $(x_n \ldots x_1)$
        where $x_i = 0$, for all $i \in I(S,D)$;
Sort all node representations and link representations in $F$ separately;
For $j \in J(S,D)$ do
/* Determine if $Q_m^{(j)}$ is fault-free. */
(i) Perform binary search for $Q_m^{(j)}$ with $*$ replaced with 0 in the sorted faulty node list.
/* Determine if there is a faulty link between $Q_m^{(j)}$ and $Q_m(S,D)$ */
(ii) Perform binary search for $(Q_m(S,D), j)$(or $(Q_m^{(j)}, j)$ if bit $j$ of $Q_m(S,D)$ is not 1) with $*$ replaced with 0 in the faulty link list.
If for some $l \in J(S,D)$, both searches fail,
then $Q_m^{(l)} \leftarrow Q_m(S,D)$.
$P(S,D) = l \circ I(S,D) \circ l$
```

In **Algorithm 1** sorting takes $O(|F| \log |F|)$ and n searches take $O(n \log |F|)$. Complexity is dominated by the complexity of procedure n-Route. Hence it is $O(n \log^2 |F|)$. We summarize the results in the following theorem.

Theorem 3.1 *In an n-dimensional hypercube Q_n, let F be the set of its faulty components. If $|F| < n$, then for any two nodes S and D, a path $P(S,D)$ with $|P(S,D)| \leq H(S,D) + 2$ can be found in $O(n \log^2 |F|)$.*

When $H(S,D) = n-1$ and S is isolated in $Q_{n-1}(S,D)$, the shortest path is of length $n+1$. Therefore the diameter of $Q_n - F$ is at most $n+1$ when $|F| < n$.

4 The Number of Faults is more than $n-1$ But Less Than $2n-2$

4.1 Finding a Path between Two Nodes

When there are at least n faults, the existence of an m-subcube such that $Q_m^{(l)} \leftarrow Q_m(S,D)$ is not guaranteed. Besides, the fault pattern can be very peculiar in that there not be any path of length $H(S,D)+2$ as shown in the next example. We assume S and D are not isolated in the rest of the paper.

Example
In a 4-dimensional hypercube, let faults occur at the corners with a 'x', as shown in Fig. 3. For S and D designated in Fig. 3, it is easy to see that the shortest path between S and D has length 6, even though the distance between S and D is 2.

In general, this scenario can happen in n-dimensional hypercube with n faults. An example can be constructed by adding more adjacent

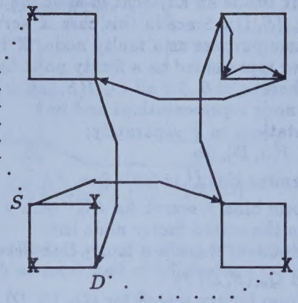


Figure 3: A worst case fault pattern which makes the shortest path between S and D of length 6 is shown above. The length of the shortest path is increased by 4.

2-subcubes $Q_2^{(l)}(\bar{S}_l, \bar{D}_l)$ to $Q_2(S, D)$ in Fig. 3 and in each of the adjacent 2-subcube $Q_m^{(l)}$ let $S \not\rightarrow \bar{S}_l$ or $\bar{D}_l \not\leftarrow D$ with at least one $S \rightarrow \bar{S}_l$ and one $\bar{D}_l \leftarrow D$ to exclude the case that S or D is isolated. With $n+m-1$ faults, the worst case scenario for S and D with $H(S,D) = m$ can also be constructed similarly such that $|P(S,D)| \geq H(S,D) + 4$. An example is shown in Fig.7. □

In the following, we describe how to route from S to D with path length $|P(S,D)| \leq H(S,D) + 4$, when $n \leq |F| < 2n-2$. Greedy approach is first used to move S and D as close to each other as possible. Starting from S, pick i such that $S \rightarrow \bar{S}_i$ and $H(\bar{S}_i, D) = H(S,D) - 1$. Repeat the process with $\bar{S}_i$ and so on, until some node is reached whose neighbors leading to decreasing the Hamming distance are all inaccessible. Perform the same procedure with D. More precisely the steps are shown in **Algorithm 2** below.

```
Algorithm 2
Sort F; /*Faulty node list and faulty link list*/;
Compute $I(S,D)$;
$P_1(S,D) = \emptyset$; $P_2(S,D) = \emptyset$;
$I = I(S,D)$; $X = S$;
Repeat
    pick $i \in I$ such that $X \rightarrow \bar{X}_i$;
    $P_1(S,D) = P_1(S,D) \circ i$
    $I = I - \{i\}$; $X = \bar{X}_i$;
until for all $i \in I$, $X \not\rightarrow \bar{X}_i$ or $I = \emptyset$;
$Y = D$;
Repeat
    pick $i \in I$ such that $\bar{Y}_i \leftarrow Y$;
    $P_2(S,D) = i \circ P_2(S,D)$
    $I = I - \{i\}$; $Y = \bar{Y}_i$;
until for all $i \in I$, $\bar{Y}_i \not\leftarrow Y$ or $I = \emptyset$;
If $I \neq \emptyset$ then /* Note $I(X,Y) = I$ */
    $P(S,D) = P_1(S,D) \circ$ "Route(X,Y)" $\circ P_2(S,D)$;
else $P(S,D) = P_1(S,D) \circ P_2(S,D)$;
```

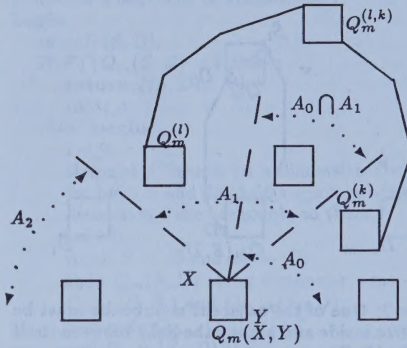


Figure 4: $Q_m(X,Y)$ together with its first level adjacent m-subcube partitioned into 3 sets, A_0 where the neighbors of S are accessible, A_1 where the neighbors of D are accessible, and A_2 where each element contains at least 2 faults inside and among the links between itself and $Q_m(X,Y)$.

It remains to design "Route(X,Y)". Let $H(X,Y) = m$. Both X and Y are isolated in $Q_m(X,Y)$, otherwise $H(X,Y)$ can be further reduced by traversing a neighbor in $Q_m(X,Y)$ of X or Y. The following Lemma 4.1 is the central result of this section.

Lemma 4.1 *$X, Y \in Q_n$, $H(X,Y) = m > 0$, and X, Y are isolated in $Q_m(X,Y)$ but not in Q_n. Let F denote the set of faults in Q_n.*
If $|F| < 2n-2$, then there exists a path, $P(X,Y)$, with length $|P(X,Y)| \leq H(X,Y) + 4$.

proof:
If one of the m-subcube, $Q_m^{(l)}$, adjacent to $Q_m(X,Y)$ is $Q_m^{(l)} \leftarrow Q_m(X,Y)$, a path $P(X,Y)$ can be found through $Q_m^{(l)}$. Therefore $|P(X,Y)| = m+2$.
In the following, we can assume that all m-subcube $Q_m^{(l)}$ adjacent to $Q_m(X,Y)$ are $Q_m^{(l)} \not\leftarrow Q_m(X,Y)$. We draw Fig. 4 to facilitate the proof. An m-subcube adjacent to $Q_m(X,Y)$ along dimension l, denoted as $Q_m^{(l)}$, is called the 1st level m-subcube. An m-subcube adjacent to two 1st level m-subcubes $Q_m^{(l)}$ and $Q_m^{(k)}$, denotes as $Q_m^{(l,k)}$, is called a second level m-subcube. Let

$A_0 = \{Q_m^{(l)} |$ Inside $Q_m^{(l)}$ and among the links between $Q_m^{(l)}$ and $Q_m(X,Y)$ there is exactly one fault and $X \rightarrow \bar{X}_l\}$

$A_1 = \{Q_m^{(l)} |$ Inside $Q_m^{(l)}$ and among the links between $Q_m^{(l)}$ and $Q_m(X,Y)$ there is exactly one fault and $\bar{Y}_l \leftarrow Y\}$

$A_2 = \{Q_m^{(l)} |$ Inside $Q_m^{(l)}$ and among the links between $Q_m^{(l)}$ and $Q_m(X,Y)$ there are at least two faults $\}$

Case I. $m \geq 2$.

If $A_0 \bigcap A_1 \neq \emptyset$, then there exists a $Q_m^{(l)} \in A_0 \bigcap A_1$, in which both $X \rightarrow \bar{X}_l$ and $\bar{Y}_l \leftarrow Y$,i.e., $\bar{X}_l$ and $\bar{Y}_l$ are accessible. It is easy to bypass a fault in $Q_m^{(l)}$. Hence, there exists a path $P(X,Y)$ with $|P(X,Y)| = m+2$.

Assume $A_0 \bigcap A_1 = \emptyset$. Further for every $Q_m^{(l)} \in A_2$, two faults can happen to make $X \rightarrow / \ \bar{X}_l$ and $\bar{Y}_l \not\leftarrow Y$. That is, in the worst case in every first level m-subcube $Q_m^{(l)}$, either $X \not\rightarrow \bar{X}_l$ or $\bar{Y}_l \not\leftarrow Y$. Hence, in every $(m+1)$-subcube $Q_m^{(l)} \cup Q_m(X,Y)$, either X or Y is isolated, and $|P(X,Y)| = m+2$ is not possible. $P(X,Y)$ has to go through a second level m-subcube $Q_m^{(l,k)}$ adjacent to $Q_m^{(l)} \in A_0$ and $Q_m^{(k)} \in A_1$. Note $|A_0| + |A_1| = n - m - |A_2|$, since there is no fault-free m-subcubes $Q_m^{(l)} \leftarrow Q_m(X,Y)$.

We claim if $|F| < 2n-2$, there must be a second level m-subcube $Q_m^{(l,k)}$ that is **Fault-Free Inside** itself, and among the links **Between** $Q_m^{(l)}$ and $Q_m^{(l,k)}$ and **Between** $Q_m^{(k)}$ and $Q_m^{(l,k)}$, where $Q_m^{(l)} \in A_0$ and $Q_m^{(k)} \in A_1$. Therefore, $P(X,Y)$ can go through $\bar{X}_l$, $Q_m^{(l,k)}$, $\bar{Y}_k$ with length $H(X,Y)+4$. We call such a 2nd-level m-subcube **IBB Fault-Free**(or simply IB^2FF).

If $|A_0||A_1| \neq 0$, there are $|A_0||A_1|$ candidates for IB^2FF m-subcube.

$$|A_0||A_1| \geq (|A_0| + |A_1| - 1) = (n - m - |A_2| - 1)$$

If $|A_0||A_1| = 0$, say $|A_0| = 0$, then there must be a $Q_m^{(l)} \in A_2$ such that $X \rightarrow \bar{X}_l$ since X is not isolated, and the number of IB^2FF candidates is at least

$$|\{Q_m^{(l)}\}||A_1| = |A_1| = (n - m - |A_2|)$$

Note as both X and Y are isolated in $Q_m(X,Y)$, $Q_m(X,Y)$ has at least $m + \delta(m > 2) \cdot m$ faults, where $\delta(m > 2) = 1$ if $m > 2$ and $\delta(m > 2) = 0$ if $m \leq 2$. So $|A_0| = 0$ and $|A_1| = 0$ can not happen, since this implies $|A_2| = n - m$ and

$$|F| \geq m + \delta(m > 2) \cdot m + 2(n - m) = \begin{cases} 2n & m > 2 \\ 2n-2 & m = 2 \end{cases}$$

a contradiction with $|F| < 2n-2$. The number of IB^2FF candidates can be injured by faults is at most

$$\begin{aligned} & |F| - |F \textstyle\bigcap Q_m(X,Y)| - |A_0| - |A_1| - 2|A_2| \\ & = |F| - |F \textstyle\bigcap Q_m(X,Y)| - (n-m) - |A_2| \\ & \quad \text{because} |A_0| + |A_1| + |A_2| = n - m \\ & \leq |F| - (m + \delta(m > 2) \cdot m) - n + m - |A_2| \\ & \quad \text{because} |F \textstyle\bigcap Q_m(X,Y)| \geq m + \delta(m > 2) \cdot m \\ & \leq n - 3 - \delta(m > 2) \cdot m - |A_2| \\ & \quad \text{because } |F| \leq 2n - 3 \end{aligned}$$

by letting every m-subcube in A_2 have exactly two faults. If $m > 2$, there are more IB^2FF candidates than the ones injured by faults, since

$$\begin{aligned} & n - 3 - m - |A_2| \\ & < \begin{cases} n - m - |A_2| - 1 & \text{if } |A_0||A_1| \neq 0 \\ n - m - |A_2| & \text{if } |A_0||A_1| = 0 \end{cases} \end{aligned}$$

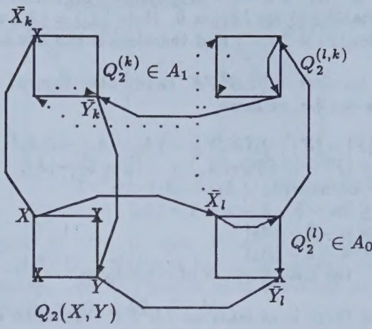


Figure 5: A worst case scenario when $m = 2$. Nodes not accessible by S or D are indicated by a cross 'x'.

Suppose $Q_m^{(l,k)}$ is IB^2FF, then

$$P(X,Y) = l \circ k \circ P(\bar{X}_{l,k}, \bar{Y}_{l,k}) \circ l \circ k.$$

Hence $|P(X,Y)| = m+4$.

However, when $m = 2$, the largest number of IB^2FF candidates injured by faults, $n - 3 - |A_2|$, can be equal to the least number of IB^2FF candidates $n - 2 - |A_2| - 1$. This happens when the following conditions are true (1)$|F| = 2n-3$, (2)$|A_0||A_1| \neq 0$ and $|A_0| = 1$ or $|A_1| = 1$, (3) for every m-subcube $Q_m^{(l)} \in A_2$, exactly two faults occur to make $X \not\rightarrow \bar{X}_l$ and $\bar{Y}_l \not\leftarrow Y$, and (4) for every $Q_m^{(l,k)}$ there is only one fault inside $Q_m^{(l,k)}$ and among the links between $Q_m^{(l,k)}$ and $Q_m^{(l)}$ or $Q_m^{(k)}$.

In this case, a subgraph as shown in Fig. 5 always exists for some l and k. As depicted by a dotted arrow line and a solid arrow line in Fig. 5, there are two node disjoint paths between $\bar{X}_l$ and $\bar{Y}_k$. Therefore there must be a fault-free path between them since only one fault can occur.

Case II. m=1.

This is the case when the link between X and Y is faulty. If $|A_0||A_1| \neq 0$, it can be verified that the number of IB^2FF candidates is

$$\begin{aligned} & |A_0||A_1| - |A_0 \textstyle\bigcap A_1| \\ & = |A_0||A_1| - (|A_0| + |A_1| - |A_0 \cup A_1|) \\ & = |A_0||A_1| - |A_0| - |A_1| + (n - 1 - |A_2|) \\ & \quad \text{because } |A_0 \cup A_1| + |A_2| = n - 1 \\ & = (|A_0| - 1)(|A_1| - 1) + n - 2 - |A_2| \\ & \geq (n - 2 - |A_2|) \\ & \quad \text{because } |A_0| \geq 1 \text{ and } |A_1| \geq 1 \end{aligned}$$

If $|A_0|$ is zero, then there exists a $Q_1^{(l)} \in A_2$ where $X \rightarrow \bar{X}_l$ and the number of IB^2FF candidates is

$|\{Q_1^{(l)}\}| \cdot |A_1| = n-1-|A_2|$. Similar argument can be established for $|A_1| = 0$. Note $|A_0| = |A_1| = 0$ implies $|F| = 2n-1$ and therefore it can not happen.
The number of IB^2FF candidates injured by faults can be, at most,

$$\begin{aligned}
&|F| - |F \bigcap Q_1(X,Y)| - |A_0 \cup A_1| - 2|A_2| \\
&= |F| - |F \bigcap Q_1(X,Y)| - (n-1) - |A_2| \\
&\quad \text{because} |A_0 \cup A_1| + |A_2| = n-1 \\
&\leq 2n-3-1-n+1-|A_2| \\
&= n-3-|A_2| \\
&< n-2-|A_2| \\
&\quad \text{the least number of candidates}
\end{aligned}$$

Hence there is at least an IB^2FF 1-subcube and a path of length 5 can go through it. □

Likewise, procedure Route(X,Y) is also constructed according to the proof Lemma 4.1. Route(X,Y) searches for a 1-st level m-subcube $Q_m^{(l)} \leftarrow Q_m(X,Y)$ or a m-subcube in $A_0 \bigcap A_1$ when $m > 1$. If both searches fail, it then searches for a second level IB^2FF m-subcube. If the third search also fails, then m must be 2. It then chooses one of the two node disjoint paths as shown in Fig. 5. Since there are not many objects to sort and search, the complexity is of $O(|F| \log |F|)$.

```
Procedure Route($X$,$Y$);
begin
make a copy of $F$ into $F'$;
For $f \in F'$ do
(i) $f = (b_n \dots b_1)$ a node, set $b_j = 0$, for all
$j \in I(X,Y)$.
(ii) $f = (b_n \dots b_{i+1} 1 b_{i-1} \dots b_1, i)$ a link, set $b_j = 0$
for all $j \in I(X,Y)$.
if $i \in I(X,Y)$, this implies that $f$ lies in $Q_m(X,Y)$
or one of its adjacent m-subcube.
In this case, $f$ serves the same purpose as a faulty
node in the algorithm to determine if a $Q_m^{(l)}$ is
fault-free, so $f$ is treated and represented as a
faulty node by $(b_n \dots b_1)$ where $b_j = 0$, for all
$j \in I(X,Y)$;

Sort the node and link representations in $F'$
separately;
Separate $F$ and sort them according to the
action on their corresponding element in $F'$;
For $j \in J(X,Y)$ do /*Search for an adjacent
$Q_m^{(j)}$ such that $Q_m^{(j)} \leftarrow Q_m(X,Y)$. */
begin
/* Determine if $Q_m^{(j)}$ is fault-free.*/
(i)Perform binary search for $Q_m^{(j)}$ with $*$ replaced
with 0 in the sorted faulty node list of $F'$.
/* Determine if there is a faulty link between
$Q_m(X,Y)$ and $Q_m^{(l)}$. */
(ii) Perform binary search for $(Q_m(X,Y), j)$(or
$(Q_m^{(j)}, j)$ if bit $j$ of $Q_m(X,Y)$ is not 1), '*'
in $Q_m(X,Y)$(or $Q_m^{(j)}$) are replaced by 0,
in the sorted list of faulty links of $F'$;
/*construct $A_0, A_1, A_0 \bigcap A_1, A_2$ and the
fault-free list.*/
If both binary searches fail, put $Q_m^{(j)}$ in the
fault-free list.
If two binary searches find only one fault, use
sorted $F$ to determine
    If $X \rightarrow \bar{X}_j$ and $\bar{Y}_j \leftarrow Y$, put $Q_m^{(j)} \in A_0 \bigcap A_1$,
    If $X \rightarrow \bar{X}_j$ and $\bar{Y}_j \not\leftarrow Y$, put $Q_m^{(j)} \in A_0$,
    If $X \not\rightarrow \bar{X}_j$ and $\bar{Y}_j \leftarrow Y$, put $Q_m^{(j)} \in A_1$,
    If at least two faults found in the two binary
    searches, put $Q_m^{(j)} \in A_2$
end;
If fault-free list is not empty,
let $P(X,Y)$ go through an $m$-subcube in it.
else if $m > 1$ and $A_0 \bigcap A_1$ is not empty,
let $P(X,Y)$ go through an $m$-subcube in it.
else /*Find a 2nd level $IB^2FF$ m-subcube.*/
Repeat
choose the $Q_m^{(l,k)}$ adjacent to both m-subcubes of
the next pair $\in (A_0 \times A_1)$ ;
(or when $|A_0| = 0$ next pair in $\{Q_m^{(l)}\} \times A_1$ where
$Q_m^{(l)} \in A_2$ with $X \rightarrow \bar{X}_l$)
    /*Determine if $Q_m^{(l,k)}$ is fault-free.*/
    Perform binary search for $Q_m^{(l,k)}$ with $*$ replaced
    with 0 in the sorted faulty node list of $F'$.
    /*Determine if there is a faulty link between
    $Q_m^{(l,k)}$ and $Q_m^{(l)}$. */
    Perform binary search for $(Q_m^{(l,k)}, l)$(or $(Q_m^{(l)}, l)$
    if bit $l$ of $Q_m^{(l,k)}$ is not 1), '*' in $Q_m^{(l,k)}$
    (or $Q_m^{(l)}$) are replaced by 0, in the sorted list
    of faulty links of $F'$.
    /*Determine if there is a faulty link between
    $Q_m^{(l,k)}$ and $Q_m^{(k)}$. */
    Perform binary search for $(Q_m^{(l,k)}, k)$(or $(Q_m^{(k)}, k)$
    if bit $k$ of $Q_m^{(l,k)}$ is not 1), '*' in $Q_m^{(l,k)}$
    (or $Q_m^{(k)}$) are replaced by 0, in the sorted
    list of faulty links of $F'$.
If all three binary searches fail, $Q_m^{(l,k)}$ is $IB^2FF$.
until $Q_m^{(l,k)}$ is $IB^2FF$ or $A_0 \times A_1(\{Q_m^{(l)}\} \times A_1)$
is exhausted.
If an $IB^2FF$ is found, then
    route through $X$,$\bar{X}_l$,$Q_m^{(l,k)}$,$\bar{Y}_k$,$Y$
else /*$m$ must be 2 and $|F| = 2n-3$
by Lemma 4.1*/
    pick a $Q_m^{(l)} \in A_0$ and a $Q_m^{(k)} \in A_1$, and choose
    a fault-free path as depicted in Fig. 5;
end;
```

The complexity of finding a second level IB^2FF m-subcube is $O(n \log |F|)$, even though $|A_0||A_1|$ can be in $O(n^2)$. This is due to the largest number of IB^2FF candidates faults can injure is $O(n)$ by Lemma 4.1. Complexity of Route(X,Y) is $O(|F| \log |F|)$ dominated by sorting faults. Therefore the complexity of **Algorithm 2** is $O(n^2 \log |F|)$ dominated by the greedy approach part of the algorithm. We summarize the results in the following theorem.

Theorem 4.1 *In Q_n, let F denote the set of faults. If $|F| < 2n-2$ and S and D are not isolated, then there exists a path $P(S,D)$ with $|P(S,D)| \leq H(S,D)+4$ and $P(S,D)$ can be found in $O(n^2 \log |F|)$.*

proof:
S and D can be moved as close to each other as possible as shown in **Algorithm 2** to node X and Y respectively. X and Y clearly satisfy the conditions of Lemma 4.1. Therefore, there exists a path of length at most $H(X,Y)+4$ between X and Y. Hence a path of length at most $H(S,D)+4$ exists between S and D by the way that X and Y are obtained in the algorithm. □

Theorem 1 in [6] is an immediate corollary of Theorem 4.1 and is stated below.

Corollary 4.1 *In Q_n, if $|F| < 2n-2$ and no node is isolated, then $Q_n - F$ is connected.*

4.2 $|F| < 2n-2$, Diameter of $Q_n - F$ is $n+2$

In [6], the author gave several shortcomings and inaccuracies of network fault-tolerance measured by edge- and vertex- connectivity. The author then proposed the concept of **forbidden faulty set**. That is certain subsets of components (e.g. processors) are considered as forbidden faulty set - they do not become faulty at the same time. Based on the concept of forbidden faulty set, some generalized measures of networks were proposed such as generalized connectivity, diameter, etc. For details, see [6].

In this section, we obtain the generalized measure of the diameter of $Q_n - F$ by assuming the neighbors of a node do not become inaccessible to the node at the same time – the forbidden faulty set. As a consequence of Theorem 4.1, the diameter of $Q_n - F$, denoted as $Dia(Q_n - F)$, is at most $n+4$. But in fact, when the distance of two nodes is n or $n-1$, a shorter path can be found as stated in the following lemma. Therefore the upper bound of network diameter $Dia(Q_n - F)$ can be further reduced.

Lemma 4.2 *In Q_n, let the set of faults be denoted by F, and S and D be two nonisolated nodes. When $|F| < 2n-2$, the following two statements are true.*
(a) If $H(S,D) = n$, then there exists a path $P(S,D)$ with length n.
(b) If $H(S,D) = n-1$, then there exists a path $P(S,D)$ with length at most $n+1$.

proof:
The statements are obviously true for $n < 4$. Assume $n \geq 4$ in the following.
(a) Consider two $(n-1)$-subscubes $Q_{n-1}(S,\bar{D}_1)$ and $Q_{n-1}(\bar{S}_1,D)$ split along dimension 1 as shown in Fig. 6. Let F_1 denote $F \bigcap Q_{n-1}(S,\bar{D}_1)$, F_2 denote $F \bigcap Q_{n-1}(\bar{S}_1,D)$ and $F_3 = F - F_1 - F_2$, i.e., the faulty links between $Q_{n-1}(S,\bar{D}_1)$ and $Q_{n-1}(\bar{S}_1,D)$. Let $|F_1| = f_1$, $|F_2| = f_2$ and $|F_3| = f_3$. It is clear that F_1, F_2, and F_3 are mutually disjoint, so $f_1 + f_2 + f_3 = |F| < 2n-2$. Without loss of generality, we can assume $f_1 \geq f_2$. (Otherwise switch S and D, and redefine all the terms.) It is clear that $f_2 \leq \lfloor \frac{2n-3}{2} \rfloor = n-2$.

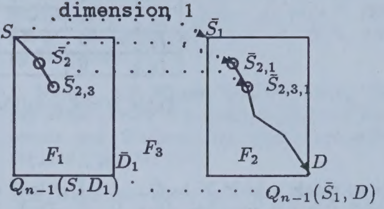


Figure 6: Q_n is split into $Q_{n-1}(S,\bar{D}_1)$ and $Q_{n-1}(\bar{S}_1,D)$. Without loss of generality, we assume $Q_{n-1}(S,\bar{D}_1)$ has at least the same number of faults as $Q_{n-1}(\bar{S}_1,D)$ does.

If $S \rightarrow \bar{S}_1$, by Lemma 3.1, a minimal path of length $n-1$ between $\bar{S}_1$ and D exists since $f_2 \leq n-2$. We may assume $S \not\rightarrow \bar{S}_1$.

S is not isolated so there is a neighbor, without loss of generality say $\bar{S}_2$, such that $S \rightarrow \bar{S}_2$. To reach a node X in $Q_{n-1}(\bar{S}_1,D)$, S can go through a neighbor of itself or a neighbor of $\bar{S}_2$. To block S such that S can not reach any node in $Q_{n-1}(\bar{S}_1,D)$ in at most 3 hops, we need $2n-2$ faults since there are $2n-2$ nodes adjacent to S or $\bar{S}_2$. Hence it is clear that S can reach a node X in $Q_{n-1}(\bar{S}_1,D)$ in at most 3 hops and $H(X,D)$ is at least $n-3$. If $f_2 < n-3$, by Lemma 3.1 a path from X to D of length $H(X,D)$ can be found. Therefore, a path between S and D is found by concatenating the two pieces.

We may assume $f_2 \geq n-3$. With the fact $f_2 \leq n-2$, there are six cases as shown in Table 1. Consider case 1 to 4 where $f_2 = n-3$. If the path length between S and X is 1 or 2, then $H(X,D) > n-3$, by Lemma 3.1 there exists a path between X and D of length $H(X,D)$. The statement is then proved by concatenating path $P(S,X)$ and path $P(X,D)$.

Assume the path length between S and X is 3 and without loss of generality we can let $X = \bar{S}_{2,3,1}$, i.e., the path traverses dimensions 2,3 and 1. If $\bar{S}_1$ and $\bar{S}_{2,1}$ is faulty then at most $n-4$ faults are left in $Q_{n-3}(X,D)$. Hence a minimal path between X and D exists.

Assume $\bar{S}_1$ and $\bar{S}_{2,1}$ are fault-free. Therefore the link between S and $\bar{S}_1$ and the link between $\bar{S}_2$ and $\bar{S}_{2,1}$ must be faulty.

Case 1 and case 2 are proved since there is no and only one faulty link in F_3 respectively.

Consider case 3. If $D \rightarrow \bar{D}_1$, then case 3 is proved since $f_1 = n-2 < H(S,\bar{D}_1) = n-1$. Assume $D \not\rightarrow \bar{D}_1$. The same argument established for S can also be established for D to prove that

	1	2	3	4	5	6
f_1	n	$n-1$	$n-2$	$n-3$	$n-1$	$n-2$
f_2	$n-3$	$n-3$	$n-3$	$n-3$	$n-2$	$n-2$
f_3	0	1	2	3	0	1

Table 1: Six possible cases when $n-3 \leq f_2 \leq n-2$.

D can reach a node Y in $Q_{n-1}(S, \bar{D}_1)$ in 2 or 3 hops.(1 is not possible since $D \not\rightarrow \bar{D}_1$.) If it takes k hops, $k = 2, 3$, then by the same argument established for S and knowing the only two faulty links in F_3 are between S and $\bar{S}_1$ and between $\bar{S}_2$ and $\bar{S}_{2,1}$, we can conclude that at most $n-k-1$ faults can be in $Q_{n-k}(Y, S)$. By Lemma 3.1, case 3 is proved, number of faults= $n-k-1 <$ the number of dimensions of $Q_{n-k}(Y, S) = n-k$.

Case 4 is symmetric with respect to S and D. The same argument used for S can be established for D to require 2 more faulty links in F_3, i.e. $f_3 \geq 4$. But $f_3 = 3$, case 4 is proved. By following the same line and using Lemma 3.1, case 5 and 6 can be proved similarly.

(b) When $H(S, D) = n-1$, we can assume S and D have the same value in the first bit. Split Q_n into $Q_{n-1}(S, D)$ and $Q_{n-1}(\bar{S}_1, \bar{D}_1)$. If $Q_{n-1}(S, D)$ has less than $n-1$ faults, a path between S and D exists. Assume $Q_{n-1}(S, D)$ has at least $(n-1)$ faults, and the rest of the cube has at most $n-2$ faults.

If $S \rightarrow \bar{S}_1$(or $\bar{D}_1 \leftarrow D$), reduce to (a) – routing from $\bar{S}_1$ to D(or S to $\bar{D}_1$). Hence a path of length $n+1$ can be found.

Assume both $S \not\rightarrow \bar{S}_1$ and $\bar{D}_1 \not\leftarrow D$. If S passes through dimension i, and 1, and D passes through dimension $k, k \neq i$ and 1, to reach a node in $Q_{n-1}(\bar{S}_1, \bar{D}_1)$, then by lemma 3.1 a minimal path between $\bar{S}_{i,1}$ and $\bar{D}_{k,1}$ can be found, because the faults making $S \not\rightarrow \bar{S}_1$ and $\bar{D}_1 \not\leftarrow D$ are not in the subcube $Q_{n-1}(\bar{S}_{i,1}, \bar{D}_{k,1})$. (Therefore at most $n-4$ faults can be in the $(n-3)$-subcube $Q_{n-1}(\bar{S}_{i,1}, \bar{D}_{k,1})$.) Note $i \neq k$, and such i, k can be found because it requires $(2n-2)$ faults to force $i = k$.

Assume either S or D has to take 3 hops to reach $Q_{n-1}(\bar{S}_1, \bar{D}_1)$. Without loss of generality, say S passes through dimension $i, j, 1$, $i \neq j$ and $i \neq 1, j \neq 1$, to reach $Q_{n-1}(\bar{S}_1, \bar{D}_1)$. Forcing S to take 3 hops, we need n faults – 1 for making $S \not\rightarrow \bar{S}_1$ and $n-1$ faults for making either $S \not\rightarrow \bar{S}_l$ or $\bar{S}_l \not\rightarrow \bar{S}_{l,1}$, $2 \leq l \leq n$. Together with the assumption that $\bar{D}_1 \not\leftarrow D$, there are left with only $n-4$ faults to block $n-1$ path of length 2 from D to a node in $Q_{n-1}(\bar{S}_1, \bar{D}_1)$. There are 3 fault-free paths of length 2 from D. Therefore there must be a $k \neq i, j$, in which dimension one of the 3 paths passes through. Now by Lemma 3.1 once again, a minimal path from $\bar{S}_{i,j,1}$ to $\bar{D}_{k,1}$ exists, since their Hamming distance is $n-4$ and at most $n-5$ faults can be in $Q_{n-4}(\bar{S}_{i,j,1}, \bar{D}_{k,1})$. (The faults making $S \not\rightarrow \bar{S}_1, \bar{S}_i \not\rightarrow \bar{S}_{i,1}, \bar{D}_1 \not\leftarrow D$ are not in $Q_{n-4}(\bar{S}_{i,j,1}, \bar{D}_{k,1})$. Note $\bar{S}_i \not\rightarrow \bar{S}_{i,1}$, othewise S could take 2 hops only.) Therefore a path of length $n+1$ is found and the path passes through dimension 1 twice.

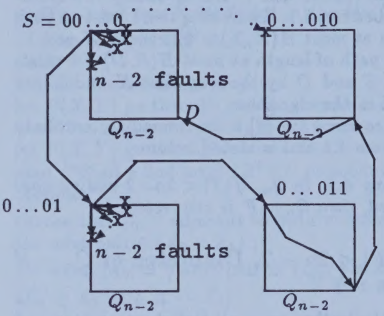


Figure 7: A worst case scenario to show that the diameter is $n+2$ when $|F| = 2n-3$.

There are not enough faults to force both S and D to take 3 hops, since that requires $2n$ faults. Hence (b) is proved. □

Theorem 4.2 *In Q_n, let F denote the fault set. If $|F| < 2n-2$ and no node is isolated by faults, the diameter of $Q_n - F$ is at most $n+2$.*

proof:
For S and $D \in Q_n$, if $H(S, D) \leq n-2$, by theorem 4.1, a path from S to D with length at most $n+2$ exists. If $H(S, D) > n-2$, by Lemma 4.2, a path of length $< n+2$ can be found. □

In the following, we construct a worst case scenario in which the shortest path between a pair of nodes with distance $n-2$ is $n+2$.

Example:
In a n-dimensional hypercube Q_n, let $S = 0^n$ and $D = 1^{n-2}0^2$, where i^n denotes k consecutive i's, $i = 0, 1$. Let the fault set be $F = \{$all neighbors of S except $0^{n-1}1$ $\} \bigcup \{$ all neighbors of $0^{n-1}1$ except S and $0^{n-2}1^2$ $\}$, as shown in Fig. 7. $|F| = 2n-3$. Clearly, a path from S to D has length at least $n+2$ as depicted in Fig 7. By Theorem 4.1, there is a path of length at most $n+2$. Therefore we can conclude the shortest path from S to D is of length $n+2$.□

From the above example, the diameter bound found in Theorem 4.2 is tight. If all the neighbors of a node are allowed to become inaccessible at the same time, then $Q_n - F$ can become disconnected in the worst case. When this happens, $Q_n - F$ has two connected components: a single node v and $Q_n - (F \cup \{v\})$. For this case our results, Theorem 4.1 and Lemma 4.2, also imply that the diameter of $Q_n - (F \cup \{v\})$ is at most $n + 2$.

5 Conclusions

In this paper, bounds on shortest paths and diameter of connected faulty hypercubes are obtained for number of faults less than $2n - 2$, where n is the number of dimensions. When number of faults is less than n, an efficient algorithm is given with complexity $O(n \log^2 |F|)$ to find a path between two nodes with length at most their Hamming distance plus 2, where $|F|$ is the number of faults. When $n \leq |F| < 2n-2$, the shortest paths between two nonisolated nodes is shown to be bounded by their Hamming distance plus 4. An algorithm is given with complexity $O(n^2 \log |F|)$ to find such a path. The diameter of connected hypercubes is bounded by $n + 2$ provided that no node is isolated and $|F| < 2n - 2$. Worst case examples are constructed to show these bounds are tight. Best known previous bounds for the diameter of connected faulty hypercubes with $|F| < 2n - 2$ is $n + 6$ [6]. We are currently working on the general case with more number of faults. Bounds on shortest path and diameter of each connected component of $Q_n - F$, when the hypercube is disconnected by faults, are also under investigation.

Acknowlegement The first author would like to thank Wei-Ming Lin for helpful discussion which led to improvement of the paper.

References

[1] B. Becker and H.-U. Simon, "How robust is the n-cube?," Proc. ACM Symp. Theory of Computing, 1986, pp. 283-291.

[2] R. Boppana and C.S. Raghavendra, "Optimal self-routing of linear-complement permutations in hypercubes," Distributed Memory Comput. Conference, 1990.

[3] G.Chartrand and L. Lesniak, "Graphs and digraphs," Wadsworth, Inc., Belmont, CA, 1986.

[4] M.-S. Chen and K.G. Shin, "On relaxed squashed embedding of graphs into a hypercube," SIAM J. Comput. vol.18, Dec. 1989, pp.1226-1244.

[5] M.-S. Chen and K.G. Shin, "Processor allocation in an n-cube multiprocessor using Gray code," IEEE Trans. on Computers, vol.36, Dec. 1987, pp.1396-1407.

[6] A.-H. Esfahanian, "Generalized measures of fault-tolerance to n-cube network," IEEE Trans. on Computers, vol. 38, Nov. 1989, pp.1586-1591.

[7] F. Harary, J.P. Hayes, and H.-J. Wu, "Survey of the theory of hypercube graphs," Computers and Mathematics with Applications 15(1988), pp. 277-289.

[8] M.S. Krishnamoorthy and B. Krishnamurthy, "Fault Diameter of Interconnection Networks," Computers and Mathematics with Applications vol.13, no. 5/6, 1987, pp. 577-582.

[9] S.L. Johnsson, "Communication efficient basic linear algebra computations on hypercube architectures," J. Parallel and Distributed Computing 4 (1987), pp.133-172.

[10] S.L. Johnsson and C.-T. Ho, "Optimal broadcasting and personalized communication in hypercubes," IEEE Trans. on Computers, vol. 38, Sept. 1989, pp. 1249-1268.

[11] T.-C. Lee and J.P. Hayes, "One-step-degradable fault tolerant hypercube," Distributed Memory Comp. Conf., 1990.

[12] W.-M. Lin and V.K.P. Kumar, "Efficient histogramming on hypercube SIMD machines," Computer Vision, Graphics, and Image Processing 49, 104-120(1990).

[13] D. Nassimi and S. Sahni, "Data broadcasting in SIMD computers," IEEE Trans. on Computers, vol. 30, Feb. 1981.

[14] F. Ozguer and C. Aykanat, "A reconfiguration algorithm for fault tolerance in hypercube multiprocessor," Inf. Processing Letters, 29(1988), pp.247-254.

[15] Y.Saad and M.H. Schultz, "Topological properties of hypercubes," IEEE Trans. on Computers, vol. 37, Jul 1988, pp.867-872.

[16] C.L. Seitz, "The cosmic cube," Comm. of Assoc. for Comput. Machinery, 28(1), 1985, pp.22-33.

[17] M.A. Sridhar, "The structure of faulty hypercubes," manuscript, Jun. 1989.

[18] S.-B. Tien, C.S. Raghavendra and M.A. Sridhar, "Reconfiguring embedded task graphs in faulty hypercubes by automorphisms," Proc. Inter. Conf. on Inf. and System Sciences, Jan. 1990.

[19] L.G. Valiant and J. Brebner, "Universal schemes for parallel communication," Proc. ACM Symposium on Theory of Computing, 1981.

Reconfigurable Embeddings of a Multi-Dimensional Mesh in a Hypercube

Thang Nguyen Bui
Usha Prabhu

Computer Science Department
The Pennsylvania State University
University Park, Pa. 16802

Abstract

We discuss reconfigurable embeddings of wrapped-around multi-dimensional grids in a binary hypercube. We look for embeddings which have small expansion and dilation, can recover from multiple sequential faults, and can be reconfigured fast using local knowledge only. We show the following.

- We present an expansion-2, dilation-1 embedding of a grid in a hypercube such that it can reconfigure itself around any single fault in one step while maitaining dilation one. Furthermore, this embedding can recover from upto $\lceil d_n/2 \rceil$ faults (with restrictions), where d_n is the largest dimension of the grid.
- We prove that *any* dilation-1, expansion-2 embedding of a grid in a hypercube can be guaranteed to recover from at most one arbitrary fault (while maintaining dilation one).
- We prove that *any* dilation-1, expansion-2 embedding of a grid in a hypercube can be guaranteed to recover from at most two arbitrary faults if (i) only the process running on the faulty processor may be moved and (ii) the dilation after reconfiguring is at most 3. We present a fast embedding algorithm satisfying the above conditions which can recover from 2 faults with high probability.
- We present an embedding of a 2^n node multi-dimensional grid in a $(n+1)$-cube which can recover from 2^n faults. The dilation of the embedding after reconfiguring is at most 5.

1. Introduction

An embedding of a guest graph $G = (V_G, E_G)$ into a host graph $H = (V_H, E_H)$ is a one-to-one mapping ϕ from V_G to V_H along with a specification of paths in H connecting the endpoints of each edge of G. We measure the quality of an embedding ϕ with three cost functions: dilation, expansion and congestion. The *dilation* of an edge $(u, v) \in E_G$ is the length of a shortest path in H that connects $\phi(u)$ and $\phi(v)$. The dilation of an embedding is the maximum dilation, over all edges in E_G. The *expansion* of an embedding is $|V_H|/|V_G|$. The *congestion* of an edge is the worst case sharing of an edge in the host by paths of the guest. The dilation of an embedding measures the worst-case distance between neighbouring elements of the guest, the expansion measures the relative size of the host.

Embedding problems can model many interesting problems, such as simulating one parallel architecture on another, assigning processes to processors in a distributed/parallel processing system, and simulating one data structure by another. A survey of embedding results is presented in [MS]. However, all these embedding techniques break down in the presence of faults in the host. We would like to reconfigure the embedding in the event of a fault in the host, thus allowing the simulation to continue. In this paper we discuss reconfigurable embeddings of $n-$dimensional grids, where each dimension is a power of two, in a binary hypercube. In the rest of this paper the term "grid" or "mesh" refers to a mesh which is wrapped-around in every dimension.

An n-dimensional grid of size $2^{d_1} \times 2^{d_2} \times \cdots \times 2^{d_n}$ can be embedded in a d-cube, where $d = d_1 + d_2 + \cdots + d_n$ with dilation 1 [SS]. If the mapping assigns node i of the grid to node H_i of the hypercube then H_i is called the *image* of i. The dilation 1 mapping can be done easily by assigning d_i bits to each dimension. Let $N_k(p)$ represent the p-th element of a k bit circular binary reflected Gray code. Then the mesh element with coordinates $(c_1, c_2, \ldots, c_n)$ denoted by $A(c_1, c_2, \ldots, c_n)$ is assigned the hypercube address $N_{d_1}(c_1)||N_{d_2}(c_2)|| \cdots ||N_{d_n}(c_n)$, where $||$ is the concatenation operation. Thus we assign neighbouring elements of the mesh to neighbouring nodes of the hypercube. This embedding uses all the nodes of the hypercube, and the system cannot be reconfigured in the presence of a fault.

However, if we use a larger hypercube of degree $d + 1$ as the host graph, then 2^d nodes remain as spares which can be used to adjust the mapping in the presence of faults so that the multi-dimensional grid can still be maintained. In [CLT], Chen et al discuss an embedding of a multi-dimensional grid in a hypercube which can recover from one arbitrary fault within a single step. This embedding will not allow the hypercube to recover from multiple faults.

In this paper we discuss reconfigurable embeddings of wrapped-around multi-dimensional grids in a binary hypercube. We assume that the size of each dimension of the grid is a power of two, and that d_n, the largest dimension of the mesh, is greater than two (i.e., we are not interested in embedding an n-cube into another n-cube). We look for embeddings which have the following properties: (i) small expansion and dilation (before and after reconfiguring), (ii) can recover from multiple sequential faults , (iii) can be reconfigured fast, i.e., no complex calculation or search has to be done to find a new configuration in the event of a fault, (iv) reconfiguration must be done using only local knowledge, i.e., a node needs only be aware of the status of nodes in its immediate neighbourhood, and (v) move as few nodes as possible. We show the following results:

- We present an embedding of a wrapped-around multi-dimensional grid of size 2^d in a hypercube of size 2^{d+1} such that if there is a fault in any node of the hypercube we can reconfigure it in one step and maintain dilation one. Furthermore, this dilation one embedding can recover from upto $\lceil d_n/2 \rceil$ faults (with restrictions), where d_n is the largest dimension of the mesh. The dilation of the embedding after reconfiguration is still one.

Assuming that faults are independent and occur with equal probability, the algorithm recovers from the second fault with probability $\approx 1 - 1/2^{d_n+1}$.

- We prove that *any* embedding of a wrapped-around multi-dimensional grid in a hypercube with unit dilation and expansion two can be guaranteed to recover from at most one arbitrary fault (while maintaining dilation one) even if the reconfiguration strategy could use unlimited time, have global knowledge and move a large number of nodes. In this sense our embedding strategy is optimal.
- We present an algorithm which can recover from an arbitrary fault while moving fewer nodes than the above algorithm. In fact this algorithm moves only the faulty node. The dilation after reconfiguring is at most 3. Assuming that faults are independent and occur with equal probability, the algorithm recovers from the second fault with probability $1 - ((4n - 1)/(2^{d+2} - 2))$. The algorithm can also recover from upto 2^{d-1} faults (with restrictions). We also prove that any unit dilation, expansion two embedding of a grid in a hypercube can be guaranteed to recover from at most two faults if (i) only the process running on the faulty processor may be moved, and (ii) it must maintain a dilation of at most three.
- If we were allowed to use larger dilation could we guarantee to recover from multiple arbitrary faults? The answer is yes, if we assume that the spare nodes do not fault. We present a dilation 1 embedding of a 2^d node wrapped-around multidimensional mesh in a $(d+1)$-cube which can recover from upto 2^d faults (with no restrictions). The dilation of the embedding after reconfiguring is at most 5. This means that every node in the initial embedding of the mesh may fail in any arbitrary pattern, and the system will still recover from the faults.

2. Failure Model

The following definitions are used:

- A hypercube node H_i is in *active* state if it is one of the image nodes of the guest graph and is fault-free. It is in *faulty* state if it is faulty. If none of the above is true, then it is in *spare* state.
- Let node j be on the right of node i along the longest dimension of the mesh. Let H_j be the image of j and H_i the image of i. These two nodes are said to be related via a parent-child relationship. We call H_i the *parent* and H_j the *child* and say that H_i is responsible for the fault of H_j. The parent keeps enough of its childs state information to allow the child process's operation to be restarted if necessary. The parent periodically checks the status of its child. If it detects that the child is faulty, it triggers a reconfiguration algorithm which attempts to recover from the fault. The recovery actions are assumed to be fault-free. We use $H_i \rightarrow H_j$ to represent this parent-child relationship.
- The links are fault-free.

3. Reconfiguration Strategy

We will first discuss the embedding of a loop (ie a 1-dim mesh) in a hypercube.

Let vertices i, j, k be three consecutive nodes of the loop. Let these be mapped (using a dilation 1 embedding) to the hypercube nodes H_i, H_j, H_k respectively. There exists a node H_δ so that $\text{XOR}(H_i, H_j, H_k, H_\delta) = 0$ and H_δ is adjacent to H_i and H_k. Assume $H_i \rightarrow H_j$ and $H_j \rightarrow H_k$. Each parent has the work environment and the code of its child. When H_j fails, node H_i can detect this fact and take the following actions:

(1) Compute $\text{XOR}(H_i, H_j, H_k)$ to get the code of H_δ which is adjacent to it. We call the node H_δ the *alternate image* of node H_j.

Node	Alternate Image
$N_{k-3}(2i-1) \Vert 000$	$N_{k-3}(2(i-1)) \Vert 100$
$N_{k-3}(2i-1) \Vert 100$	$N_{k-3}(2i-1) \Vert 010$
$N_{k-3}(2i-1) \Vert 110$	$N_{k-3}(2i-1) \Vert 101$
$N_{k-3}(2i-1) \Vert 111$	$N_{k-3}(2i) \Vert 110$
$N_{k-3}(2i) \Vert 111$	$N_{k-3}(2i-1) \Vert 011$
$N_{k-3}(2i) \Vert 011$	$N_{k-3}(2i) \Vert 101$
$N_{k-3}(2i) \Vert 001$	$N_{k-3}(2i) \Vert 010$
$N_{k-3}(2i) \Vert 000$	$N_{k-3}(2(i+1)-1) \Vert 001$

Table 1. The nodes of a loop labelled using a k-bit instant spare code and their unique alternate images which are not part of the loop.

(2) Reset the node H_δ by sending it the backup work environment of H_j, and the code of its child H_k.

If we can guarantee that node H_δ is a spare node, then the system has reached a consistent valid state so that it can resume operation. The loop will still be embedded in the hypercube with dilation 1. In the next section, we show how we can guarantee that H_δ is a spare node. A similar method to embed a loop in a hypercube has been developed independently in [Me] and [PM].

Initial mapping

Let us define an *instant spare* Gray code M_k using k bits and containing 2^{k-1} elements as follows. Let $M_k(p)$ be the p-th element of a k bit instant spare Gray code, and $N_k(p)$ be the p-th element of a k bit circular binary reflected Gray code. Let us denote the alternate image of $M_k(p)$ by $\widehat{M_k}(p)$.

The instant spare Gray code can be generated as follows:

```
$j = 0$
for $i = 1$ to $2^{k-4}$ do
        $M_k(j+1) = N_{k-3}(2i-1) \| 000$
        $M_k(j+2) = N_{k-3}(2i-1) \| 100$
        $M_k(j+3) = N_{k-3}(2i-1) \| 110$
        $M_k(j+4) = N_{k-3}(2i-1) \| 111$
        $M_k(j+5) = N_{k-3}(2i) \| 111$
        $M_k(j+6) = N_{k-3}(2i) \| 011$
        $M_k(j+7) = N_{k-3}(2i) \| 001$
        $M_k(j+8) = N_{k-3}(2i) \| 000$
        $j = j + 8$
next $i$
```

Let us call the leading $k-3$ bits of $M_k(p)$ the *prefix* and the last 3 bits the *suffix* of the code.

Theorem 1. *Let a loop (i.e., a 1-dim mesh) of length $2^{k-1}, k > 3$, be labelled using a k bit instant spare Gray code. Then the alternate image of every node in this loop is unique, and is not part of the instant spare Gray code.*

Proof: We note that for any $i = 1$ to 2^{k-4} the only elements in M_k with prefix $N_{k-3}(2i-1)$, end with the one of the following: $000, 100, 110, 111$ i.e., any $M_k(p)$ that has an odd element of N_{k-3} as its prefix has one of $000, 100, 110, 111$ as its suffix. It cannot have $001, 010, 011$, or 101 as its suffix. Similarly, any $M_k(p)$ that has an even element of N_{k-3} as its prefix has one of $000, 001, 011$ or 111 as its suffix.

However, from Table 1 we note that all the alternate images which have an odd element of N_{k-3} as its prefix have one of 001, 010, 011 or 101 as their suffix. Also any alternate image which has an even element of N_{k-3} as its prefix has one of 010, 100, 101 or 110 as its suffix. Therefore the alternate images do not form part of the instant spare Gray code. From Table 1, we can also see that every node has a unique alternate image described by this Gray code. ■

We can now define the initial mapping of the mesh into the hypercube. Let n be the largest dimension of the mesh. By our assumption, d_n is greater than two. Assume d_i bits are used to encode dimension i for all dimensions 1 to $n-1$ using a circular binary-reflected Gray code as explained above. $d_n + 1$ bits are used to encode dimension n using the instant spare Gray code.

Theorem 2. *Let a mesh be embedded in a hypercube using the above mapping. Then the alternate image of every active node is in a spare state, and the alternate image of any active node is different from the image of any other.*

Proof: The mesh node $Y = A(i_1, i_2, \ldots, i_n)$ has the image $Y' = N_{d_1}(i_1) \parallel N_{d_2}(i_2) \parallel \cdots \parallel N_{d_{n-1}}(i_{n-1}) \parallel M_{d_n+1}(i_n)$ (which is an *active* node of the hypercube). Its parent is the image of $X = A(i_1, i_2, \ldots, i_n - 1)$ and its child is the image of $Z = A(i_1, i_2, \ldots, i_n + 1)$. The alternate image of Y' has the code $N_{d_1}(i_1) \parallel N_{d_2}(i_2) \parallel \cdots \parallel N_{d_{n-1}}(i_{n-1}) \parallel \widehat{M}_{d_n+1}(i_n)$. But by Theorem 1 this node has not been used in the embedding and is therefore a spare node. Moreover, every active node of the hypercube has a *unique* spare node as its alternate image. ■

4. Algorithms for Reconfiguration

Using the initial configuration described earlier, we can describe the following algorithms to handle faults. Note that the alternate image of any node is always described with reference to the initial configuration. We represent the image (using our initial embedding) of a mesh node Y by Y' and its alternate image by Y''.

Algorithm 1

Assume a fault occurs in the image of node $Y = A(i_1, i_2, \ldots, i_n)$. Its parent X' (the image of $X = A(i_1, i_2, \ldots, i_n - 1)$) detects the error. We know that Y has the image $Y' = N_{d_1}(i_1) \parallel N_{d_2}(i_2) \parallel \cdots \parallel N_{d_{n-1}}(i_{n-1}) \parallel M_{d_n+1}(i_n)$. If any neighbour of Y' or Y' itself had (a) faulted earlier, or (b) been moved to its alternate image, then the system cannot recover from the fault. Otherwise, we move all active nodes Z' which have $M_{d_n+1}(i_n)$ as their least significant $d_n + 1$ bits to their alternate images. We also notify all their neighbours that they have moved.

The dilation and congestion of the new mapping is still 1. A total of 2^{d-d_n} nodes are moved to their alternate images whenever a fault occurs. Note that we cannot maintain dilation one if only the faulty node is allowed to be moved.

This algorithm can only handle one error occuring in the images of every alternate line of the n-th dimension of the mesh. This is because the parent and child of an active node Y' are adjacent to Y''s alternate image. However, the alternate image of Y''s parent and the alternate image of Y' are not adjacent. Thus, if faults occured in nodes $N_{d_1}(i_1) \parallel N_{d_2}(i_2) \parallel \cdots \parallel N_{d_{n-1}}(i_{n-1}) \parallel M_{d_n+1}(i_n)$ and $N_{d_1}(i_1) \parallel N_{d_2}(i_2) \parallel \cdots \parallel N_{d_{n-1}}(i_{n-1}) \parallel M_{d_n+1}(i_n + 1)$, they would both be moved to their alternate images and we would no longer have dilation 1.

Thus, this algorithm can handle upto $\lceil d_n/2 \rceil$ faults as long as no two faulty nodes are images of (a) Mesh nodes $A(i_1, i_2, \ldots, i_{n-1}, i_n)$ and $A(j_1, j_2, \ldots, j_{n-1}, i_n)$ respectively or of (b) mesh nodes $A(i_1, i_2, \ldots, i_{n-1}, i_n)$ and $A(j_1, j_2, \ldots, j_{n-1}, i_n \pm 1)$ respectively. The congestion of the embedding is, of course, one. ■

We now show that we cannot do much better than this, i.e., we can pick two faults in the hypercube such that some circular mesh can no longer be embedded in the hypercube with dilation one.

Scott and Brandenburg [SB] have shown that in any dilation one embedding of a mesh in a hypercube, each coordinate direction must use a separate set of bits. They also show that the bit that is different when moving from node j to node $j+1$ in the ith coordinate is independent of the values of the other coordinates. It is obvious that each dimension of length 2^i will require i bits.

Assume we are embedding a mesh of size 2^m in a hypercube of size 2^{m+1}. Assume that there is a fault in node 1^{m+1} and in node 0^{m+1}. We show that for any value of m, $m > 4$, we can construct a mesh which cannot be embedded in the hypercube.

Pick the 2-dimension mesh of size $2^2 \times 2^{m-2}$. Pick the bits used by the smaller dimension. This dimension may use two or three bits.

If it uses only two bits, we obviously cannot find an embedding since we cannot find a Gray code of length 4 using 2 bits when two of the choices (11 and 00) are not allowed.

Assume 3 bits have been allotted for this dimension. Using the 6 allowable choices (001, 011, 010, 110, 100, 101) we must create a closed Gray code sequence of length 4.

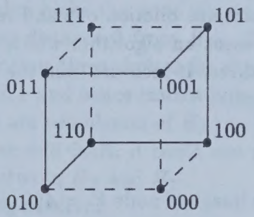


Figure 1. Faults in nodes 000 and 111 result in no available "faces" in the 3-cube.

We know that if both i and j are two different neighbours of a node k in a hypercube, then there is a unique node which is adjacent to both i and j – this is $i \oplus j \oplus k$. Scott and Brandenburg call this the square property. This says that to get a closed Gray code sequence of length 4 we need a "face" or square of the hypercube. But in Figure 1 we see that because of the faults, such a face does not exist.

Therefore this mesh cannot be embedded in the hypercube with 2 faults. Using a similar argument, we can show that a $2 \times 2^{m-1}$ mesh, $m > 3$, cannot be embedded in an $m+1$-cube with faults at nodes 0^{m+1} and at 1^{m+1}.

In general it is not enough to say that deleting two nodes with distance $m+1$ causes there to be no m-dimensional subcube, and therefore no grid of size 2^m can be embedded in the cube. Our algorithm can embed a $2^3 \times 2^{m-3}, m > 5$, grid in a 2^{m+1}-cube with dilation one even if nodes 0^{m+1} and 1^{m+1} (distance $m+1$) are faulty.

From this we get the following theorem:

Theorem 3. *Any scheme to embed an arbitrary wrapped-around mesh of size* $2^{d_1} \times 2^{d_2} \times \cdots \times 2^{d_n}, d_n > 2$, *in a hypercube of size* 2^{d+1} *(where* $d = d_1 + d_2 + \cdots + d_n$ *) can guarantee to recover from at most one arbitrary fault.*

We have thus shown that our scheme is optimal in that there exist meshes and fault sequences which do not allow the mesh to be embedded in the hypercube with 2 faults.

Assuming a model where faults are independent and uniform, this algorithm will recover from a single fault with probability 1. It will not recover from a second fault in two cases:

1. If the first fault was on a spare node, and the second fault was on an active node sharing the same least significant $d_n + 1$ bits as the first fault.
   The probability of this happening is
   $$\frac{1}{2} \cdot \frac{2^{d-d_n-1}}{2^{d+1}-1}$$
2. The scheme cannot handle a second fault if the first was on an active node and the second was on another active node with the same least significant $d_n + 1$ bits or a neighbour of one of these nodes along dimension n of the mesh. The probability of this happening is
   $$\frac{1}{2} \cdot \frac{2^{d-d_n-1} + 2 \cdot 2^{d-d_n-1}}{2^{d+1}-1}$$

Therefore, the probability of recovering from a second fault is

$$1 - \frac{2^{d-d_n}}{2^{d+1}-1}$$

or at least 0.935 (for $d = 4, d_n = 3$), becoming 0.99 for $d = 7, d_n = 6$.

The disadvantage of this algorithm is that multiple nodes are moved in the event of a fault. The advantage is that it maintains dilation one and recovers from the second fault with very high probability. We now present an algorithm which moves only the faulty node while maintaining a dilation of at most three. It recovers from the second fault with high probability and can recover from upto 2^{d-1} faults.

Algorithm 2

Assume a fault occurs in the image of node $Y = A(i_1, i_2, \ldots, i_n)$. Its parent (the image of node $X = A(i_1, i_2, \ldots, i_{n-1})$) detects the error. We know that Y has the image $Y' = N_{d_1}(i_1) \parallel N_{d_2}(i_2) \parallel \cdots \parallel N_{d_{n-1}}(i_{n-1}) \parallel M_{d_n+1}(i_n)$. If (a) the image of Y had faulted earlier or if (b) the images of either neighbour X or Z of Y along dimension n of the mesh had faulted earlier or (c) the images of one of the neighbours of X or of Z along dimensions $1 \ldots n-1$ of the mesh had faulted earlier, then we cannot recover from the fault. Otherwise move Y' to its alternate image Y'' and notify all its neighbours that it has moved. We now have an embedding with dilation at most 3.

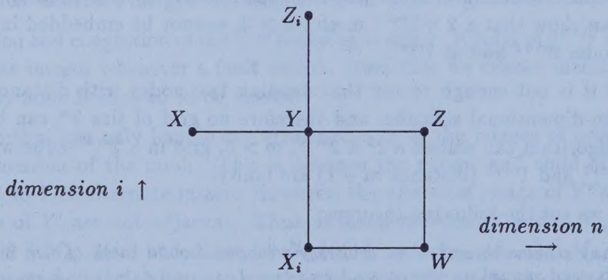


Figure 2. A node Y of a mesh and its relevant neighbours.

To see this we observe that if X, Y, Z were neighbours along dimension n of the mesh then X',Y'',Z' are connected with links of length 1 (Figure 2). However in all other dimensions this is not true. If X_i,Y,Z_i were neighbours along dimension i of the mesh, then X_i',Y'' are distance 3 apart, and so are Y'',Z_i'. The communication path used between X_i' and Y'' is X_i',W', Z',Y'' where W' is X_i's child and Z's neighbour along dimension i. By the conditions checked earlier these nodes are fault-free.

Thus this algorithm can recover from 2^{d-1} faults as long as (a) an active node and its alternate image do not become faulty or (b) neither neighbour X' or Z' of a faulty node Y' along dimension n of the mesh become faulty and (c) none of the neighbours of X' or of Z' along dimensions $1 \ldots n-1$ of the mesh become faulty. ∎

We show that no algorithm which handles faults dynamically and fast (i.e., only the process running on the faulty processor is moved) can guarantee to recover from more than 2 faults with dilation 3. We allow the algorithm to take its recovery actions after the first random fault – let this fault be at a the hypercube node H_X, the image of the mesh node X. The algorithm must reconfigure by moving the process at H_X to a free node at a distance of 1 or 2 from H_X.

If it picks a node H_Y at distance 1, then there is a neighbour Z of X such that H_Z is distance 2 from H_Y. There are exactly 2 paths of length 2 or 3 from H_Z to H_Y, one of which passes through H_X and cannot be used. Let an adversary pick the intermediate node on the other path between H_Z and H_Y to become faulty. The algorithm cannot recover from this fault since there is now no path of length 2 or 3 between H_Z and H_Y.

Now assume that the algorithm picks the node H_Y at distance 2 from H_X. Let Z be a neighbour of X such that H_Z is at distance 3 from H_Y. Three bits must be changed when moving from H_Z to H_Y. There are thus three choices for the first hop in the path between the two nodes. One of these choices is H_X and hence cannot be used. An adversary could then pick the other two choices (these nodes are neighbours of H_Z) to become faulty. Regardless of how the algorithm recovered from the second fault, it could not recover from the third fault since there is now no path of length 3 between H_Z and H_Y.

From this we get the following theorem:

Theorem 4. *Any dynamic scheme to embed a wrapped-around mesh of size* $2^{d_1} \times 2^{d_2} \times \cdots \times 2^{d_n}$, $d_n > 2$, *in a hypercube of size* 2^{d+1} *(where* $d = d_1 + d_2 + \cdots + d_n$ *) with unit dilation can guarantee to recover from at most two arbitrary faults if (i) only the process running on a faulty processor may be moved and (ii) the dilation after reconfiguring is at most three.*

Our algorithm is almost optimal in that it can recover from two faults with high probability. The advantage of our algorithm is that all paths between vertices are uniquely determined and need not be found dynamically. Also it can recover from upto 2^{d-1} faults.

Assuming that faults occur independently and with equal probability, we can calculate the probability that the algorithm will recover from the second fault. This algorithm cannot recover from a second fault in two cases:

1. An active node and its alternate image become faulty (in any order). The probability of this happening is
$$\frac{1}{2} \cdot \frac{1}{2^{d+1}-1}$$
2. An active node becomes faulty, followed by the images of either of its two neighbours X and Z along dimension n of the mesh, or the images of one of the neighbours of X or of Z along dimensions $1 \ldots n-1$ of the mesh. The probability of this is
$$\frac{1}{2} \cdot \frac{2+2(2(n-1))}{2^{d+1}-1}$$

Therefore the probability that Algorithm 2 can handle two faults is
$$1 - \frac{4n-1}{2^{d+2}-2}$$

At the very minimum this is 0.887 for n=2 and d=4, becoming 0.944 for n=2, d=5 and is 0.99 for n=2, d=8. We note that this algorithm performs better than algorithm 1 if d_n is small compared to d (i.e., all dimensions of the mesh are approximately the same size)

Algorithm 3

We can recover from more errors while maintaining a dilation of 5 if we assume that spare nodes cannot become faulty. Without such an assumption *no* algorithm can recover from more than $d + 1$ faults (even if it could use unlimited dilation), if we allow only the faulty node to be moved (use a reasoning similar to the proof of Theorem 4 to prove this, except make the neighbours of H_Y faulty). Also such an assumption allows us to compute paths between nodes very fast, using local information only. Assume a fault occurs in the image of node $Y = A(i_1, i_2, \ldots, i_n)$. Its parent (the image of node $X = A(i_1, i_2, \ldots, i_{n-1})$) detects the error. We know that Y has the image $Y' = N_{d_1}(i_1) \parallel N_{d_2}(i_2) \parallel \cdots \parallel N_{d_{n-1}}(i_{n-1}) \parallel M_{d_{n+1}}(i_n)$. Move Y' to its alternate image Y''. We now have an embedding with dilation 5.

In the mesh of Figure 2 assume that Y' has been moved to its alternate image Y''. A path of length 1 exists between Z' and Y''. If Z' faults and Z is moved to its alternate image Z'', there is a path of length 3 between Y'' and Z''. Since all the alternate images form a d-dimension hypercube, and Y'' and Z'' differ in 3 bits and we have assumed that the alternate images do not fault, we can always find a fault-free path of length 3 from Y'' to Z'' using alternate image nodes only.

For X_i to communicate with Y a message can be sent from X_i' to W'' (using a path of length 1). From W'' send a message to Z'' (using a link of length 1). We then send a message from Z'' to Y'' as explained above. The path length is thus 5. If X_i' fails, a link of length 1 exists between X_i'' and Y''.

Therefore the maximum distance between Y'' and any of its neighbours is 5.

If we assume that faulty nodes can transmit messages, then the dilation of this embedding is 3 (messages passed from X_i' to W' to Z' to Y'' using links of length 1).

Since we can recover from a fault in any of the originally active nodes, we can recover from 2^d faults while maintaining a dilation of 5.

5. Conclusion

We have presented several fault-tolerant embeddings of a wrapped-around multi-dimensional grid in a binary hypercube. We assumed that the size of each dimension of the grid was a power of two. Using the results of [Ch], we would like to extend these results to embedding arbitrary meshes in the hypercube.

6. References

[Ch] M. Y. Chan, "Embedding of *d*-Dimensional Grids into Optimal Hypercubes", *Proc. 1^{st} Ann. ACM Symp. on Parallel Algs. and Arch.*, (June 1989), 52-55.

[CLT] S. K. Chen, C. T. Liang, W. T. Tsai, "An Efficient Multi- Dimensional Grids Reconfiguration Algorithm on Hypercubes", *Proc. of 18th Int. Symp. on Fault Tolerant Computing Systems*, (1988), 368-373.

[MS] B. Monien and I. H. Sudborough, "Comparing Interconnection Networks", *Proc 1988 Int. Math. Found. Computer Sci. Conf.*, (1988), in Springer Verlag's Lecture Notes in Computer Science, Vol 324, 138-153.

[Me] R. G. Melhem, "Bi-Level Reconfigurations of Fault-Tolerant Arrays", Technical Report 89-4, Dept of Computer Science, University of Pittsburgh, (1989).

[PM] F. J. Provost, R. Melhem, "Distributed Fault Tolerant Embedding of Binary Trees and Rings in Hypercubes", *Proc of International Workshop on Defects and Fault Tolerance in VLSI Systems*, (1989).

[SS] Y. Saad, M. H. Schultz, "Topological Properties of Hypercubes", *IEEE Transactions on Computers*, (July 1988), 867-872.

"Move and Smooth" Routing Algorithms on Mesh-connected Computers

Taisook Han Donald F. Stanat

Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, North Carolina 27599

Abstract

We describe two representatives of a class of deterministic algorithms for one-to-one and one-to-many message routing problems on square meshes. Initially, each processor contains at most one message, and each message has one or more designated destinations. No processor is the destination of more than one message.

Move and smooth algorithms are recursive. Initially, the entire mesh is considered a single region. At each recursive stage,

> Each region is partitioned into congruent subregions;
> A copy of each message is moved to each of the regions that contains one of its destinations (the *move* phase);
> Messages within each region are re-distributed so that each processor contains at most one message (the *smooth* phase).

The recursion continues until each region contains a single row or column of processors, at which time each message has arrived at or can be moved directly to its destination.

On a square n by n mesh, one of the algorithms requires $5.5n$ parallel communication steps and five buffers per processor; the other requires $9n$ parallel communication steps and three buffers per processor. These costs are not changed for a class of one-to-many routing problems.

Extended Abstract

1 Introduction

Routing algorithms on meshes have been considered by several researchers. In the earliest papers [6, 7], several routing algorithms are described for mesh-connected SIMD processor arrays such as the ILLIAC IV. Later papers [2, 3, 4, 8] describe routing algorithms for mesh-connected processor arrays with distributed control. Krizanc et al. [2] describe a randomized algorithm which requires $2n + O(\log n)$ steps to route messages with a constant number of buffers in each processor on an n by n mesh. Kunde [3] describes a deterministic routing algorithm which performs routing in $2n + O(n/f(n))$ steps with $2f(n)$ buffers, where $1 \leq f(n) < n/2$. By choosing $f(n) = 2$ for comparison with our algorithm, his algorithm will require $4n + o(n)$ steps assuming a sorting algorithm with $4n + o(n)$ time complexity is provided. Leighton et al. [4] improve the time bound of Kunde's algorithm to the optimal value of $2n-2$ using processors with a constant number of buffers. Recently, Makedon and Simvonis [5] suggest a heuristic algorithm which uses two buffers.

Our work was motivated by the desire to find routing algorithms that could be used not only for one-to-one routing problems, but also, with equal ease, for one-to-many routing problems. Many routing algorithms (e.g., those of Kunde) make substantial use of sorting and derive great advantage from it. However, sorting appears less appropriate for one-to-many routing problems. This motivated us to deliberately forego the use of sorting as a building block for routing algorithms. In this paper we describe two representatives of a new class of routing algorithms that we call *move and smooth* algorithms and show how they behave on one-to-one routing problems.

The particular algorithms we describe use only a small number of buffers per processor (three or five) regardless of the mesh size, but have time bounds considerably worse that the optimal $2n - 2$. However, if we ignore the cost of duplicating messages, the cost of these algorithms is not increased for one-to-many routing problems in which messages may be sent to several processors, but no processor receives more than one message.

Our routing algorithms have the following properties:

- The algorithms are distributed; that is, each processor operates independently on the basis of only messages in its buffers and its state. There is no global planning.
- The number of buffers required in a processor is constant regardless of the size of the mesh.
- All processors work in lock-step; no handshaking between processors is necessary.

The remainder of this paper is organized as follows. Section 2 describes the model of a processor array, and notation. Section 3 is a general description of our algorithms. Section 4 describes the two algorithms and their time complexity. Section 5 gives more detail of the smoothing step in our algorithms.

2 Definitions and Notation

We assume a two dimensional n by n square array, or grid, of n^2 processors, where $n = 2^p$. Each processor (except those on the boundaries) is connected to four other processors, called the north, south, west, and east neighbors. The connections, or links, between processors can carry a single message in each direction in a single time step; thus a processor not on the boundary can send up to four messages to its neighbors and receive up to four messages from them in a single time step. Each processor has a unique address; we denote by $P(i, j)$ the processor located at the intersection of row i and column j. Message sending between processors is synchronous, using packet switching.

A message M in a processor $P(i)$ consists of 3 fields: a list of destination addresses, a target address, and data. A *destination address* $D(M)$ is an address to which a message is finally delivered. A *target address* $T(M)$ is an intermediate address for which a message is currently heading. A new target address is computed whenever a message reaches its target; the final target address is the destination address. In the case of a one-to-many problem, a message may 'split' when it reaches a target address, resulting in two messages with the same data but different lists of destination addresses and different target addresses. In all our work, the *priority* of a message is defined to be the distance from the current position of the message to its current target address, measured by a "city block" metric.

3 Move and Smooth Algorithms

Each of our algorithms is recursive, with each recursive stage consisting of two steps. The *move* step moves each message in the array to a subpart of the mesh that contains its destination. The move step may, however, put more than one message into some processors. The *smooth* step redistributes the set of messages in each subpart of the mesh so that each processor once again contains only a single message.

Our algorithms are based on successive partitionings of the mesh into b disjoint congruent *regions*; that is, each partitioning divides a mesh (or submesh) into b disjoint regions that are identical in size and shape. We will describe one algorithm which breaks each square submesh into quarters ($b = 4$), and another which first breaks a square submesh into two halves ($b = 2$) and then breaks each half into two parts again ($b = 2$), giving quarters. During each move step, each message moves from its current region to the region that contains its destination. The move step is always devoid of contention for communication channels, but it may result in some processors containing more than one message. In particular, if the number of regions in the partition is b, a processor may be required to hold up to $b + 1$ messages during the move step, and up to b messages at the end of the move step. In order to apply the algorithm recursively, the precondition of 'at most one message per processor' must be re-established.

The purpose of the smooth step of each recursive stage is to distribute the messages of each region so that each processor contains only one message without moving any message out of the region that contains its destination. Completion of the smooth step re-establishes the precondition of the recursive algorithm that each message is in the region that contains its destination, and each processor contains at most one message. Then the algorithm recurs; each region is partitioned into disjoint subregions and each message is sent to the subregion that contains its destination. The algorithm terminates after the stage in which each region has only one row or column of processors. When each region consists of a one dimensional array of processors, each message travels to its destination without contention.

4 The Algorithms

4.1 Algorithm Q

The first algorithm is called Q because it divides square meshes into quarters, which we refer to as *quadrants*. We describe the algorithm for the special case of a one-to-one problem on a mesh of size n by n, where $n = 2^p$. Figure 1 provides an example.

The move step begins with a partitioning of the square array of processors into four quadrants of equal size. Initially, each processor has at most one message. If a message is initially in the correct quadrant (that is, the quadrant that contains its destination address), it does not travel during the move step. Otherwise, a message travels from its current location to the processor which has the same relative position in the correct quadrant. This movement is accomplished by moving 0 or $n/2$ steps along a column of the mesh, and then 0 or $n/2$ steps along a row. Thus, a message may not move, or it may travel $n/2$ steps along a row or a column, or it may travel $n/2$ steps along a column and then $n/2$ steps along a row. In any case, when the first move step is finished, each message will be in the quadrant that contains its destination processor. All messages start moving at the same time, and stop as soon as they reach the correct position in the destination quadrant. During the move step, a processor may contain up to five messages; one that did not move, and four transients. All messages move concurrently, and since there is no contention for links, the time for the first move step on an n by n mesh is n.

At the end of a move step, a processor may contain up to four messages; one from before the move step, and three that were moved there from corresponding processors in the other three quadrants. In order to recur, we must re-establish the precondition that each processor contains at most one message. (Otherwise, the number of messages in a single processor could grow exponentially with the number of recursive steps.) Distributing the messages among the processors is the task of smoothing, described later. For now, we simply note that smoothing requires less than $1.75n$ steps for an $n/2$ by $n/2$ quadrant of an n by n mesh. Smoothing ends the first stage of the recursive algorithm, and leaves each message in the quadrant that contains its destination, with at most one message per processor. Because the move step takes n steps and the smooth step takes $1.75n$ steps, the complexity of the algorithm is given by the following recurrence:

$$\begin{aligned} T(1) &= 0 \\ T(n) &\leq T(n/2) + 2.75n \quad (\text{for } n = 2^p) \end{aligned}$$

which has the solution $T(n) \leq 5.5n$. We will derive this value, and refine the expression given above, in Section 5.

4.2 Algorithm H

The second algorithm we treat, which we call H (for 'Halves'), has a somewhat worse time complexity than that of algorithm Q, but requires only three buffers in each processor. For the special case of a one-to-one problem on a square mesh, algorithm H first partitions the mesh into two rectangular regions, each of which is half the mesh. Move and smooth steps are performed on those two (non-square) regions, after which every message is in the correct half of the array, and every processor has at most one message. The algorithm then partitions each of those two non-square regions into two square regions and performs move and smooth steps on them. Then the algorithm recurs. Like the first algorithm, each recursive stage of this algorithm moves each message into the correct quadrant of the current region. Algorithm H, however, requires two separate move and smooth steps for each recursive stage. During the first move step, each message will travel either 0 or $n/2$ positions along its current row; during the second move step, each message will travel either 0 or $n/2$ steps along its current column. Thus, the maximum distance travelled by any message in the first stage of this algorithm is n, the same as for algorithm Q.

Algorithm Q and H differ in the complexity of their smoothing steps. Specifically, the first smoothing step of the first stage of H (which divides an n by n mesh into halves) requires $2n$ steps, and the second smoothing step of the first stage of H (which divides each of the halves into quarters of the original mesh) requires $1.5n$. Thus, adding in the n steps required for the move, the complexity of the algorithm is given by

$$\begin{aligned} T(1) &= 0 \\ T(n) &\leq T(n/2) + 4.5n \quad (\text{for } n = 2^p) \end{aligned}$$

giving a solution of $T(n) \leq 9n$.

5 Smoothing

Both algorithms Q and H rely on *smoothing*, which takes a rectangular submesh which may have several messages in a processor and distributes the messages so that no processor contains more than one message. Each submesh that undergoes smoothing is a member of a partition into b subsets; thus,

it is a rectangular mesh with r rows and c columns,
initially, no processor contains more than b messages, and
the total number of messages in the submesh is no greater than rc.

For convenience, we shall assume that r and c are both even integers.

Smoothing is done in two phases. The *counting* phase informs each processor about the initial and final configurations of messages. After the counting phase, each processor knows the number of messages that it will receive from and send to its neighbors to accomplish smoothing. The *distribution* phase moves each message in the mesh to achieve the final configuration. During the distribution phase, processors transmit and receive messages along the row and then transmit and receive messages along the column. (Messages are moving either along a row or along a column at any time in a mesh, but not both.) After the distribution phase, each processor will have at most one message.

5.1 Counting

We will use the row-major index of processors in the rectangle as their relative address. Thus, for a rectangle with r rows and c columns, a relative address of a processor at (r', c') is given by $r'c + c'$ where $0 \leq r' < r$ and $0 \leq c' < c$.

To speed up the process of smoothing, we divide the mesh into halves (top and bottom) and perform counting and distribution on the two halves simultaneously and independently. If there are u messages in the top half before smoothing, and v messages in the bottom half, then after smoothing, each processor with index 0 through $u - 1$ and each processor with index $rc - v$ through $rc - 1$ will contain a single message, while the remaining processors will be empty. Note that u or v may be greater than the number of processors in the half; excess messages in either half are moved into the other half of the mesh. But no processor need to contain more than one message after smoothing, because $u + v \leq rc$. In the following, we describe only the algorithm for counting and moving messages in the upper half of the mesh; the algorithm for the lower half of the mesh can be constructed by a simple transformation on the processor indices.

We denote by $C(r', c')$ the number of messages in the processor (r', c') before the counting phase; $0 \leq C(r', c') \leq b$. The counting phase provides information which will be used to determine the movement of messages during distribution. Specifically, counting provides to each processor (r', c') in the upper half of the mesh, where $0 \leq r' < r/2$ and $0 \leq c' < c$, the following:

1. the total number of messages in row r', $N_{r'}$, that is, $\sum_{i=0}^{c-1} C(r', i)$,
2. the number of messages to the left of processor (r', c') in row r', $L_{r',c'}$, that is, $\sum_{i=0}^{c'-1} C(r', i)$,
3. the total number of messages in rows above row r', $S_{r'-1}$, that is, $\sum_{i=0}^{r'-1} \sum_{j=0}^{c-1} C(i, j)$.

If a mesh is a single column or row, there is no need to count, because messages can move to their destination processor without any contention. If a mesh contains more than one column and row, and r is divisible by two, the following procedure is used.

In the following, we represent count information from eastern, western and northern neighbors by E, W, and N respectively. The number of messages contained in a processor is denoted by Z. The input from a nonexistent neighbor or a neighbor that is not part of the mesh is defined to be 0.

1. In parallel, for each row, information flows from the left end of each row to the right of the row (from west to east), with each processor sending to its eastern neighbor the value $Z + W$. The flow begins with the leftmost processor of each row sending to its eastern neighbor the value Z. This message wave provides each processor (r', c') with the number of messages to its left in the row r'. Similarly, the processors in the rows send the value $E + Z$ to their western neighbor. From these a processor can compute $N_{r'}$, the total number of messages in its row. This requires $c - 1$ steps.

2. A second wave of message now occurs in each column, with each processor sending its southern neighbor the value of $N + E + W + Z$, which specifies the number of messages in rows above it. This wave originates in the top row, where $N = 0$. Since counting occurs independently in the top and bottom $r/2$ rows, this wave requires $r/2 - 1$ steps.

Lemma 1 *The counting phase on an r by c mesh requires $r/2 + c - 2$ steps.*

5.2 Distribution

Once the number of messages have been counted on a rectangle, distribution of the messages occurs in two phases. In the first phase, messages are rearranged in their current rows (*row movement*) so that no column of the rectangle contains more than r messages; messages are almost evenly spread in each row. In the second phase, messages travel along their current columns (*column movement*) so that no processor has more than one message. Movement along rows is completed before movement along columns begins.

Since processors initially have up to b messages, and each link can handle only one message at a time, messages compete for the links. During row movement, messages are treated as indistinguishable, so priorities are not used. For column movement, priorities are based on to what direction to move and how far.

We first describe row movement in a r by c submesh. The purpose of row movement is to distribute the messages in the submesh as evenly as possible across the columns of the submesh. This can be done by distributing the messages of each row as evenly as possible across processors if, in addition, one makes sure that, when the number of messages is not equal to the number of processors, the 'extra' messages of different rows are not placed in the same columns. Row movement places the 'extra' messages of successive rows in successive columns. Thus, if row 0 has k_0 extra messages (that is, the number of messages mod $c = k_0$), then after row movement processors 0 through $k_0 - 1$ each have one more message than the other processors. And if row 1 has k_1 extra messages, then these messages are placed in columns k_0 through $(k_0 + k_1 - 1) \bmod c$. Thus, row movement distributes the extra messages of each row evenly through the columns, so that after row movement, on an r by c mesh, the following holds:

> For any submesh consisting of the first r' rows, $0 \le r' < r/2$, either the total number of messages in each column is the same, or there exists some k, such

that the total number of messages in any column $c' < k$ is greater by one than the total number of messages in any column $c' \geq k$.

Each processor can, from the information received from the counting phase, compute the number of messages it must send out and receive from its neighbors during row movement as well as the number of messages that it must contain after row movement. Row movement is implemented by each processor sending messages in the appropriate direction as soon as they are available.

After row movement, there are at most r messages in any column of the mesh. Moreover, each processor in the upper half of the mesh can determine the number of messages in the column above, Specifically, for processor (r', c'), let $U_{r',c'}$ be the number of messages in rows 0 through r' of column c'. Then $U_{r',c'}$ is given by the following equations:

$$\begin{aligned} U_{r',c'} &= \lceil S_{r'}/c \rceil \quad \text{if } c' < S_{r'} \bmod c \\ U_{r',c'} &= \lfloor S_{r'}/c \rfloor \quad \text{if } c' \geq S_{r'} \bmod c \end{aligned}$$

Each processor (r', c') knows both $U_{r',c'}$ and $U_{r'-1,c'}$. Likewise, each processor in the lower half of the mesh can determine the number of messages in the column below. Since there must be at most one message after column movement, each processor can determine which direction and how far each of its messages should travel.

Theorem 1 *Consider a one dimensional array of processors indexed from* 0 *to* $c-1$, *where each processor contains at most* b *messages, where* b *is an even integer. Row movement requires at most* $\max(c, b(b+2)c/4(b+1))$ *time steps.*

For algorithm Q, row movement of the distribution phase in an r by c submesh takes at most $1.2c$ steps; because $b = 4$. Algorithm H requires two row movements, each of which requires at most c steps ($b = 2$).

Theorem 2 *When processors have up to* b *messages initially in an* r *by* c *rectangular mesh, column movement requires at most* $r(b-1)/b$ *steps.*

Our results are summarized as follows. For algorithm Q operating on a square n by n mesh, where n is a power of 2, the move step of the first stage of the recursion requires n *data message steps.* The smooth step requires counting and distribution. Counting requires $0.75n$ *integer message steps* in the first recursive stage. (Integer message steps involve only passing a single integer between processors.) Distribution, consisting of row and column movement in each $n/2$ by $n/2$ submesh requires at most $.975n$ *data message steps* in the first recursive stage. These values can be used to refine the recurrence equations for algorithm Q given in Section 4. In the following equations, t_M denotes the cost of sending a message packet from one processor to another, and t_m denotes the cost of sending a single integer from one processor to another.

$$\begin{aligned} T(1) &= 0 \\ T(n) &\leq T(n/2) + 2n\,t_M + 0.75n\,t_m. \end{aligned}$$

Theorem 3 *Algorithm Q on an* n *by* n *array of processors requires five buffers,* $4n$ *data message steps, and* $1.5n$ *integer message steps.*

The following recurrence equations for algorithm H are obtained similarly.

$$\begin{aligned} T(1) &= 0 \\ T(n) &\leq T(n/2) + 2.75n\,t_M + 1.75n\,t_m. \end{aligned}$$

Theorem 4 *Algorithm H on an n by n array of processors requires three buffers,* 5.5n *data message steps, and* 3.5n *integer message steps.*

The routing costs do not change for one-to-many routing problems because messages are duplicated only at target addresses during the move steps and the new copies introduce no contention.

6 Generalizations

Move and smooth algorithms can be implemented in a variety of ways, and for higher dimensions. The primary requirement is that each step partition each submesh into congruent regions. For example, algorithm H can be extended to partition an n by n mesh into b vertical strips and then b horizontal strips for each recursive step. In this case, the time requirement decreases at the cost of an increase in buffer requirements. In higher dimensions, the easiest move and smooth algorithms divide each submesh into two congruent parts. Algorithm H can also be extended to r successive partitionings of an r dimensional hypercube of size n^r into two halves along each axis.

We can improve the time bound of algorithms H by overlapping the counting and distribution phases at the cost of a more complex control structure. With this overlapping, the cost of algorithm H is reduced to $5.5n\,t_M + 2n\,t_m$. An analogous approach can reduce the cost of algorithm Q to $4n\,t_M + n\,t_m$.

References

[1] T. Han. Move and smooth algorithms on mesh-connected computers. Technical report, Dept. of Computer Science, University of North Carolina at Chapel Hill, December 1990.

[2] D. Krizanc, S. Rajasekaran, and T. Tsantilas. Optimal routing algorithms for mesh-connected processor arrays. In *VLSI Algorithms and Architectures: 3rd Aegean Workshop on Computing, AWOC 88*, pages 411–422, 1988.

[3] M. Kunde. Routing and sorting on mesh-connected arrays. In *VLSI Algorithms and Architectures: 3rd Aegean Workshop on Computing, AWOC 88*, pages 423–433, 1988.

[4] T. Leighton, F. Makedon, and I. Tollis. A 2n-2 step algorithm for routing in an $n \times n$ array with constant size queues. In *Proceedings of the 1989 ACM Symposium on Parallel Algorithms and Architectures*, pages 328–335, 1989.

[5] F. Makedon and A. Simvonis. Fast parallel communication on mesh connected machines with low buffer requirements. In *Proceedings of 1990 IEEE International Conference on Computer Design: VLSI in Computers & Processors*, pages 78–81, 1990.

[6] D. Nassimi and S. Sahni. An optimal routing algorithm for mesh-connected parallel computers. *Journal of the ACM*, 27(1):6–29, January 1980.

[7] C. S. Raghavendra and V. K. P. Kumar. Permutations on Illiac IV-type networks. *IEEE Transactions on Computers*, C-35(7):662–669, July 1986.

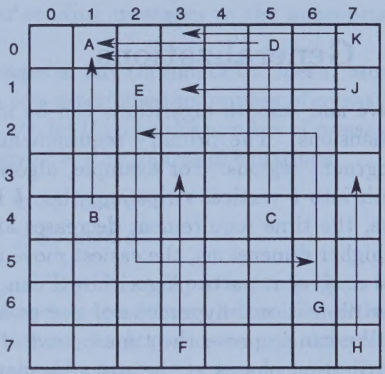
[8] S. Rajasekaran and T. Tsantilas. An optimal randomized routing algorithm for the mesh and a class of efficient mesh-like routing networks. In *7th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 226–241, 1987.

Processors in the 8 by 8 array are addressed by row-column index pairs. Then messages, labelled A through K, have destinations as specified in the Destination Table at the right.

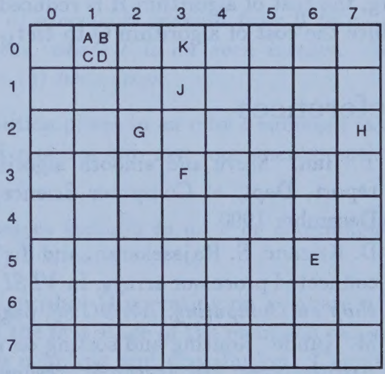
Destination Table

Message	A	B	C	D	E
Destination	2,0	2,1	2,2	2,3	6,7
Message	F	G	H	J	K
Destination	0,1	0,2	0,6	3,0	3,1

In the first phase of algorithm Q, each message travels to the correct quadrant of the mesh. The Figure at the right shows the original position of the messages and the paths followed by them during the first Move step. Note that since message A is already in the correct quadrant, it does not change position. Because all messages begin travelling simultaneously, there is no contention for links. At the end of the first Move step, processor (0,1) contains four messages.



The result of the first Move step, ready for the Smooth step, which begins with counting. Each processor receives sufficient information to determine the message flow of the row and column movements during the distribution phase. The row movement spreads messages almost evenly in each row of a quadrant so that the number of messages are almost even in each column. The messages A,B,C,D and K are spread in row 0 of the top left quadrant.



The array after the Distribution phase of the first Smooth step. The target of messages A,B,C and D are arbitrary, but their set of targets is not. They could be permuted. No target of other messages are arbitrary. Messages in the upper half are packed in low-indexed processors and messages in the lower half are packed in high-indexed processors.

The algorithm will now recur, breaking each 4 by 4 quadrant into four 2 by 2 quadrants and doing another Move and Smooth.

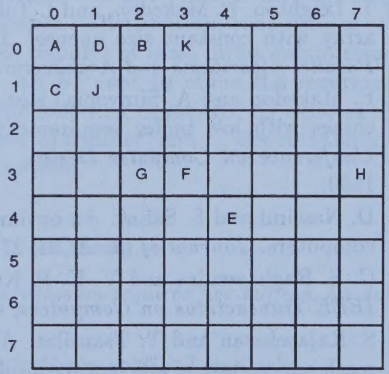


Figure 1: An example of the first recursive phase of Algorithm Q

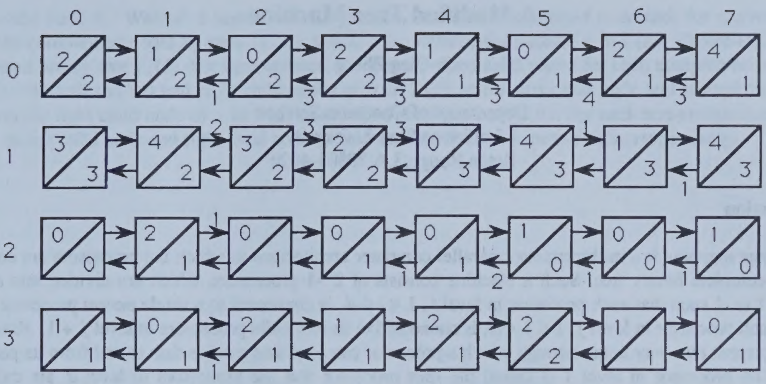


Figure shows an example of initial and final configurations of row movement for the upper half of an 8 by 8 square mesh. Processors are represented by squares; the row and column indices appear above each column and to the left of each row. The number of messages initially contained in each processor is shown in the squares as an integer above the diagonal, while the number of messages to be contained in each processor after row movement is shown below the diagonal. Arrows represent the channels between processors, and the direction of these arrows represents the direction of the channels. Integers on the arrows are the number of messages which flow through the channel. An arrow with no integer indicates that no message uses the channel during row movement. During the counting phase, each processor receives the number of messages in rows above, the number of messages in the current row and the number of message to the left in the current row. The first two numbers are used to determine the final configuration of the row. The last number is necessary to determine the message flow to achieve the final configuration. In the final configuration, the number of messages in any pair of processors in each row differs by at most one; for example, the processors $(1,1)$, $(1,2)$, and $(1,3)$ contain two messages, and the rest of processors in row 1 contain three messages. Note the column position of the processors with an extra message for each row. In the final configuration, the number of messages in each column differs only by one for any set of rows 0 through j. For example, in columns 0 through 4, the total number of messages in rows 0, 1, and 2 is five, while columns 5, 6, and 7 contain only four messages in rows 0 through 2. Note that processor $(0,3)$ contains two messages and processor $(0,4)$ contains one message in row 0, while processor $(1,4)$ contains one more message than processor $(1,3)$ in row 1.

After row movement, each processor can determine the number of messages in the column above it. This enables each processor to determine which direction and how far each of its messages should travel. During the column movement, the messages are sent to the target row. For example, processor $(2,1)$ knows there are total 4 messages in processors $(0,1)$ and $(1,1)$. Therefore, the message in processor $(2,1)$ will travel 2 steps down along the column.

Figure 2: An example of distribution phase for the upper half of an 8 by 8 mesh

A Modified Tree Machine

Si-Qing Zheng

Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803-4020

I. Introduction

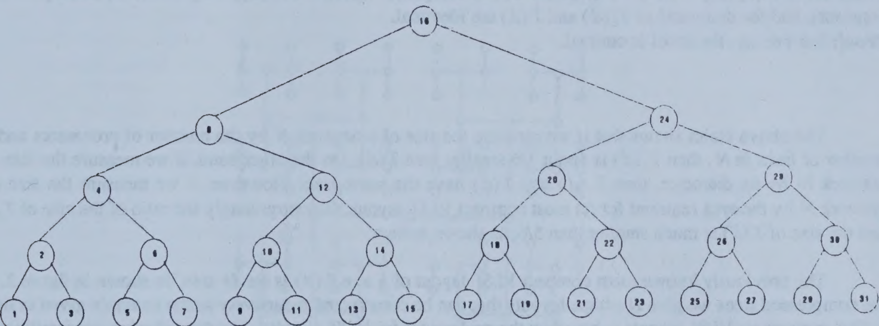
A *tree machine* is a multiprocessor parallel computer architecture in which the processors are connected to form a complete binary tree. Such a machine consists of 2^d-1 processors, which are divided into d levels, numbered 1 to d, such that each processor in level i, $1 < i \leq d$, is connected to a single *parent* processor in level i - 1 and each processor in level j, $1 \leq j < d$, is connected to its two *child* processors in level $j + 1$. Branches of the tree represent two-way links, through which a processor can send and receive data to and from its parent and children. The processor in level 1 is called the *root* processor and the processors in level d are called *leaf* processors. Usually, the root and leaf processors are the only processors with input and output capabilities. In figure 1(a), a complete binary tree of 2^5 - 1 nodes is illustrated.

As binary trees capture the essence of divide-and-conquer strategies, the tree machines support several useful logarithmic-time functions such as broadcast, search, census, and select-and-rotate, assuming that data communication along a link takes constant time ([11][19]). Designing efficient parallel algorithms for tree machines has received extensive attention (see [1][2][4][8][10][16][19] for examples). Since a tree machine has constant degree of interconnections between processors, it is more convenient to implement a tree machine on a *VLSI* chip with smaller area, compared with those networks with relatively larger number of links between processors, such as hypercubes and shuffle-exchange networks. Assuming the grid model for *VLSI* circuits, an interesting way to lay out a binary tree is called *H-tree layout*([3]). The H-tree corresponding to the complete binary tree of 2^7-1 nodes is shown in figure 2. Investigations on tree layout problems have led to many interesting results (see [5][6][7][9][13][14][20][21]).

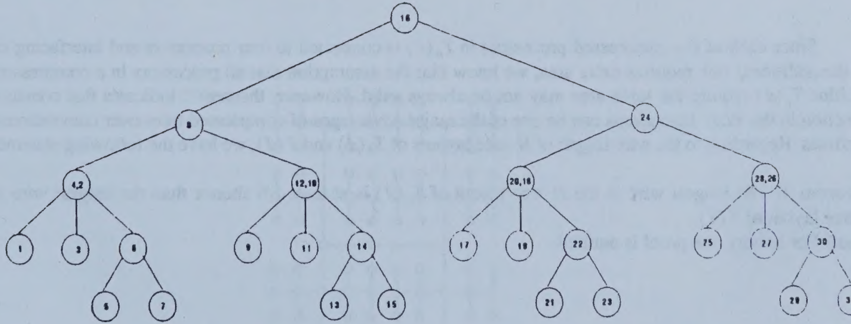
In general, the computing power and the cost of a multiprocessor system are proportional to the structural complexities of the system. The number of processors and the number of data communication links in a multiprocessor system are certainly major factors in determining the complexity of the system. In the *VLSI* implementations of multiprocessor systems, due to the physical limits (such as circuit density, area, wire length, number of conducting layers, etc.) of *VLSI* technology, reducing system complexities while maintaining the same or similar system performances is extremely important: Comparing the performances of two multiprocessor computer systems involves comparing the efficiencies of parallel algorithms for a specified set of applications. The efficiency of a parallel algorithm A for a problem P on a parallel machine M is the ratio between the product of the time complexity of A and the number of processors in M, and the time complexity of the best sequential algorithm solving problem P. In this paper, we investigate the problem of reducing the cost of a multiprocessor system M without lossing the efficiencies of algorithms on M. Specifically, we propose a modified tree architecture called *compressed tree machines*. Let $T(d)$ denote the complete binary tree of d levels and the array of processors with $T(d)$ as its underlying interconnection network, and let $T_c(d)$ denote a compressed tree corresponding to $T(d)$ and the array of processors with $T_c(d)$ as its underlying interconnection network. We use $P(T)$, $L(T)$ and $A(T)$ to denote the number of processors, the number of links and the area of the most compact *VLSI* layout of tree T, respectively. We show that $P(T_c(d)) = (5/6)*P(T(d))$, $L(T_c(d)) = (5/6)*L(T(d))$, and $A(T_c(d)) = (5/9)*A(T(d))$. We also show that most efficient parallel algorithms on a tree machine that extensively utilize the binary tree structure of the interconnection network can be slightly modified so that they perform the same on compressed tree machines. For the computation intensive applications, we present a simple load balancing method for compressed tree machines that guarantees the optimal simulation of tree machines, in terms of the speed and number of processors, for highly parallelizable applications.

II. Compressed Trees

For a given graph $G = (V,E)$, where V is the node set (note that for consistence, we use nodes to refer to vertices) and E is the edge set of G, a *contraction* of an edge $e = (u,v) \in E$ is the following operation: delete edge e and merge u with v. We define a compressed tree $T_c(d)$ as a tree constructed from $T(d)$ by the following contraction operations: for every node a in level $l = d - 2i - 1 > 1$, contract the edge connecting a and its left child node b. We call a node formed by contraction as a *compressed node* and, for convenience, we relabel the compressed nodes by pairs (a, b), where b is the left child node of a in $T(d)$. Figure 1(b) shows the compressed binary tree $T_c(5)$. For convenience, we identify three child nodes for each compressed node (a, b) in $T_c(d)$. We call the left and right child nodes of b in $T(d)$ as the first child node and second child node of (a,b) and the right child node of a in $T(d)$ as the third child node of (a,b). For each non-compressed node, we rename its left child node and right child node as its first child node and second node, respectively.


16
8
24
4
12
20
28
2
6
10
14
18
22
26
30
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31

(a) Complete binary tree T(5).


16
8
24
4,2
12,10
20,18
28,26
1
3
6
9
11
14
17
19
22
25
27
30
5
7
13
15
21
23
29
31

(b) Compressed tree $T_c(5)$.

Figure 1: Complete tree and compressed tree.

It is easy to see that the nodes in a compressed tree $T_c(d)$ are of degrees no greater than 4. All nodes of degree 1 are the leaf nodes of $T_c(d)$, and they correspond to the leaf nodes of $T(d)$. A node is a compressed node if and only if it has degree 4. The only node in $T_c(d)$ that has degree 2 is the root node. When d is odd, the two child nodes of the root of $T_c(d)$ are of degree 3. When d is even, the two child nodes of the root of $T_c(d)$ are compressed nodes. We define a compressed tree machine as a multiprocessor network whose underlying structure is a compressed tree. Compressed trees have regular structures. That is, the structure of $T_c(d)$ can be defined recursively on d. Such a property, which is desirable in designing interconnection networks, allows us to derive several interesting results in the computational and implementation aspects of compressed tree machines. Let *P(T)* and *L(T)* denote the number of processors and the number of data communication links in a tree T, respectively. The diameter of a network is the maximum of the length of shortest paths between all pairs of processors. Let us compare $T_c(d)$ with $T(d)$.

Theorem 1: $P(T_c(d)) < (5/6) * P(T(d)) + c_1$ and $L(T_c(d)) < (5/6) * L(T(d)) + c_2$, where c_1 and c_2 are small constants, and the diameters of $T_c(d)$ and $T(d)$ are identical.
Proof: For brevity, the proof is omitted.

□

The above claim shows that if we measure the size of a network N by the number of processors and the number of links in N, then $T_c(d)$ is about 1/6 smaller than $T(d)$. On the other hand, if we measure the size of a network N by its diameter, then $T_c(d)$ and $T(d)$ have the same size. However, if we measure the size of a network N by the area required for its most compact *VLSI* layout, then surprisingly the ratio of the size of $T_c(d)$ and the size of $T(d)$ is much smaller than 5/6, as shown bellow.

The previously known most compact *VLSI* layout of a tree $T(d)$ is the H-tree, as shown in figure 2. For the compressed trees we give H-shape layouts that can be constructed recursively as the example given in figure 3. Our analysis on *VLSI* layouts is based on the grid model for *VLSI* circuits. Readers who are not familiar with the design of *VLSI* circuits may refer to [19], where *VLSI* design rules introduced in [12] and abstract models of *VLSI* circuits introduced in [18] are succinctly presented. Let $W(T)$ and $H(T)$ denote the width and height of the smallest rectangle containing the H-tree layout of tree T, respectively, and let $A(T)$ denote the area of H-tree layout of tree T. Assuming that each processor in a compressed tree machine $T_c(d)$ requires the same constant area as a processor in the tree machine $T(d)$, we have the following statement.

Theorem 2: $A(T_c(d)) < (5/9) * A(T(d)) + c_3$, where c_3 is a small constant.
Proof: For brevity, the proof is omitted.

□

Since each of the compressed processors in $T_c(d)$ is connected to four processors and interfacing circuit for the additional link requires extra area, we know that the assumption that all processors in a compressed tree machine $T_c(d)$ require the same area may not be always valid. However, theorem 2 indicates that considerable reduction in the *VLSI* layout area can be one of the major advantages of compressed trees over conventional tree machines. Regarding to the wire length of H-tree layouts of $T_c(d)$ and $T(d)$, we have the following statement.

Theorem 3: The longest wire in the H-tree layout of $T_c(d)$ is at least 1/6 shorter than the longest wire in the H-tree layout of $T(d)$.
Proof: For brevity, the proof is omitted.

□

The aspect ratio of a rectangle of width W and height H is defined as $\mathbf{max}\{W, H\}/\mathbf{min}\{W, H\}$. Let us make further comparisons between the H-tree layouts of $T_c(d)$ and $T(d)$. For odd d, the aspect ratio of the smallest rectangle containing the H-tree layout of $T(d)$ is 1, whereas the aspect ratio of the smallest rectangle containing the H-tree layout of $T_c(d)$ is 5/4. For even d, the aspect ratio of the smallest rectangle containing the H-tree layout of $T(d)$ is 2, whereas the aspect ratio of the smallest rectangle containing the H-tree layout of $T_c(d)$ is 8/5. How about the smallest squares containing the H-tree layouts? Let $A^*(T)$ denote the area of the smallest square containing the H-tree layout of T. Then, we have the following statement.

Theorem 4: $A^*(T_c(d)) < (25/36) * A^*(T(d)) + c_4$ when d is odd, and $A^*(T_c(d)) < (4/9) * A^*(T(d)) + c_5$ when d is even, where c_4 and c_5 are small constants.
Proof: For brevity, the proof is omitted.

□

Theorems 2, 3 and 4 show that using H-tree layout it is possible to implement a compressed tree machine $T_c(d)$ on a *VLSI* chip more area-efficiently than the *VLSI* implementation of a tree machine $T(d)$. This suggests that compressed tree machines can be more cost-effective than conventional tree machines, considering that the *VLSI* implementation is a major aspect of the cost of a processor array.

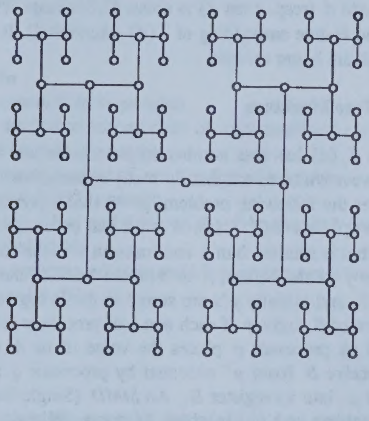


Figure 2: H-tree layout of T(7).

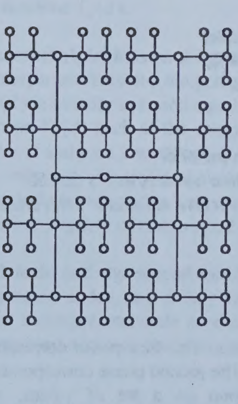


Figure 3: H-tree layout of $T_c(7)$.

Clearly, the H-tree layouts imply schemes for simulating tree machines and compressed tree machines by processor arrays of two-dimensional mesh structure. It is well known that one way to characterize the simulations between two multiprocessor networks is *graph embedding*. Let G and H be two simple undirected graphs. An *embedding* of G in H is a one-to-one mapping of the nodes of G into the nodes of H, together with a specification of paths in H connecting the images of the endpoints of each edge in G. The *dilation* of the embedding is the maximum length of these paths in H and the *congestion* of the embedding is the maximum number of edges of G whose corresponding mapped paths in H include a single edge in H. The parameters dilation and congestion are used to measure the efficiencies of simulation of the network with underlying graph G by the network with underlying graph H. Clearly, the H-tree layout of a binary tree $T(d)$ (resp. compressed tree $T_c(d)$) defines a congestion-1 embeddings of $T(d)$ (resp. $T_c(d)$) into a two-dimensional mesh. We call such embeddings as *H-tree embeddings* of $T(d)$ (resp. $T_c(d)$). Theorem 2 implies that a compressed tree $T_c(d)$ can be H-tree embedded into a two-dimensional mesh of size about 4/9 smaller than the size of a two-dimensional mesh for H-tree embedding a tree $T(d)$, and the dilation of H-tree embedding of $T_c(d)$ is 1/6 smaller than the dilation of H-tree embedding of $T(d)$ (theorem 3). Furthermore, the smallest square two-dimensional mesh for the H-tree embedding of $T_c(d)$ of odd d (resp. even d) is about 11/36 (resp. 5/9) smaller than the the smallest square two-dimensional mesh for the H-tree embedding of $T(d)$ (theorem 4). In other words, simulating $T_c(d)$ instead of $T(d)$, processors in a mesh are better utilized.

III. Performance of Compressed Tree Machines

A compressed tree machine $T_c(d)$ has less number of processors and less number of communication links than a tree machine $T(d)$. However, it turns out that for many applications, $T_c(d)$ can perform equally well as $T(d)$. To see this, let us consider the following problem: given that a positive number x stored in the root processor and 2^{d-1} positive numbers y_i, $1 \le i \le 2^{d-1}$, stored in the leaf processors of the tree machine $T(d)$, find the largest y in the leaf processors that is smaller than x and transmit it to the root. We may specify that in case there does not exist such a number y in the leaves, x is returned back to the root. Let us assume that each processor has two registers A and B, and initially y's are stored in the B registers of the leaves, x is stored in the A register of the root, and content of B register of each non-leaf processor is 0. We define that the statement "**send** A **to** processor q" executed by processor p places the value of its A register on the link connecting processor q, and the statement "**receive** B **from** p" executed by processor q stores value on the link, which would be placed there by processor p, into its register B. An *SIMD* (Single Instruction Stream Multiple Data Stream) parallel algorithm for this problem on a tree machine $T(d)$ is as follows.

```
ALGORITHM CAB_1
    for i = 1 to d - 1 do
        for all processors P do in parallel
            if P is not a leaf then send A to the two children of P;
            if P is not the root then receive A from its parent;
        endfor
    endfor
    for all processors P do in parallel
        if P is not a leaf then A := 0;
        else if B < A then A := B
    endfor
    for i = 1 to d-1 do
        for all processors P do in parallel
            if P is not the root then send A to P's parent;
            if P is not leaf then receive A from P's left child and receive B from P's right child;
            A := max {A, B};
        endfor
    endfor
end of CAB_1
```

This algorithm consists of two phases. The first phase corresponds to the broadcast function, which distribute the value x to all leaf processors. The second phase corresponds to a census function, which is defined as a commutative and associative operation on a set of values, such as finding maximum, addition, multiplication, etc. Based on the following fact, algorithm *CAB_1* can be transformed into an algorithm on the compressed tree machine $T_c(d)$ with only minor changes.

Fact 1: Let $h(q)$ denote the height of the subtree rooted at node q in $T_c(d)$. For any node p in $T_c(d)$, h(first child of p) = h(second child of p) = m, and if p is a compressed node then h(third child of p) = $m+1$.

For $T_c(d)$ with even d, the new algorithm is as follows.

```
ALGORITHM CAB_2
    for i = 1 to d-1 do
        for all processors P do in parallel
            if P is not a leaf then send B to all P's children;
            if P is not the root then receive A from P's parent;
        endfor
        for all processors P do in parallel
            if P is not a leaf then A := 0;
            else if B < A then A := B
        endfor
    endfor
    for i = 1 to d-1 do
        for all processors P do in parallel
            if P is the first or second child of some node and i is odd
            then send A to P's parent;
            if P is the third child of some node and i is even
            then send A to P's parent;
            if P is not a leaf and d is odd then receive A from P's first child and receive B from
            P's second child;
            if P is a compressed node and i is even then receive A from P's third child
            A := max {A, B};
        endfor
    endfor
end of CAB_2
```

Since the height of $T_c(d)$ is the same as that of $T(d)$ (see theorem 1), the correctness of algorithm *CAB_2* can be easily established by using fact 1. For the case that d is odd, a similar algorithm can be obtained. Comparing algorithm *CAB_2* with algorithm *CAB_1*, we notice that they have the same performance. This example shows that any parallel algorithm on tree machine $T(d)$ using a sequence of compute, aggregate and broadcast operations, such a algorithm belongs to a class of tree-based algorithms called compute-aggregate-broadcast (*CAB*) algorithms, can be converted into an equally efficient algorithm running on a compressed tree machine $T_c(d)$. Moreover, this example also shows that any algorithm whose efficiency heavily depends on the communication network structure of a tree machine $T(d)$ can be converted into a equally efficient algorithm on the the corresponding compressed tree machine $T_c(d)$.

For the computation intensive parallel algorithms on the tree machine $T(d)$ such that the communication time can be ignored, we give a rough comparison between the performances of $T_c(d)$ and $T(d)$. We assume that for an algorithm A each processor in $T(d)$ is initially assigned equal amount of computation C. Correspondingly, we assume that for the same algorithm on $T_c(d)$ each of the compressed processors is assigned a workload of $2C$ and each of the remaining processors is assigned a workload of C initially. Note that such initial workloads of processors in $T_c(d)$ can be easily enforced. Our objective is to redistribute the computation among the processors of $T_c(d)$ as even as possible and, moreover, the data communication involved during the computing process is as less as possible. The following property of compressed trees is useful for load balancing.

Fact 2: Both of the first child node and the second child node of a compressed node p are either compressed nodes or non-compressed nodes, and if they are non-compressed nodes they must be leaf nodes; the third child node q of a compressed node p must be a non-compressed node, and both two child nodes of q are either compressed nodes or non-compressed nodes, and if they are non-compressed nodes they must be leaf nodes.

Suppose that C amount of workload in each processor can be decomposed into two halves of $C/2$ each. By fact 2, for each compressed processor p, $C/2$ of its workload can be passed to its third child q, and after processor q finishes its computation, it then passes the results to p when necessary. By such a simple load balancing method, $T_c(d)$ takes 1.5 time the computing time required by $T(d)$ to complete the computation, assuming that transmitting a data item between two processors requires a small constant time.

The above load balancing method for $T_c(d)$ enforces $1.5C$ workload for a processor that is either a compressed processor or the third child of a compressed processor, while each of the remaining processors, including the leaf processors that constitute more than half of the total number of processors, maintains C workload. Now, assume that C amount of computation in each processor is ideally parallelizable, i.e. it can be evenly decomposed into five parts. Then, we have the following simple load balancing method which can be used for a compressed tree machine $T_c(d)$ to optimally simulate a tree machine $T(d)$.

```
ALGORITHM LOAD_BALANCING
    for all processors $P$ do in parallel
        if $P$ is a compressed processor then pass $C/5$, $C/5$ and $(3/5)C$ of its workload to $P$'s first,
            second and third child processors, respectively;
        if $P$ is the third child processor then pass $C/5$ of its workload to each of its two child
            processors;
    endfor
end of LOAD_BALANCING
```

This load balancing algorithm is presented in the *SIMD* fashion to emphasize that the load balancing process is synchronizable. This algorithm can be easily modified to fit an *MIMD* or a distributed computer system with $T_c(d)$ as its interconnection network, for which load distribution is carried out asynchronously by migrating processes among processors. By fact 2, we know that each processor has at most $(6/5)C$ workload after the load balancing. Since $P(T_c(d)) = (5/6)P(T(d))$, the efficiencies of the same algorithm running on $T_c(d)$ and $T(d)$ are identical. However, if the computation workload assigned to each processor is highly sequential, then compressed tree machine $T_c(d)$ is one time slower than tree machine $T(d)$, since each compressed processor in $T_c(d)$ is assigned a workload of $2C$. This is the worst case.

Now, consider any parallel algorithm A on a tree machine $T(d)$. By identifying the communication intensive parts and computation intensive parts of A, and treating these parts differently by the methods discussed above, a new algorithm A' on the compressed tree machine $T_c(d)$ can be obtained. We believe that for most applications, compressed tree machine $T_c(d)$ can simulate $T(d)$ very efficiently.

III. Discussions

We proposed a new parallel computer architecture, compressed tree machine, based on trees. This architecture modifies the tree machines by reducing the number of processors and the number of data communication links, and maintaining a regular tree structure. We showed that by slight modifications, most algorithms utilizing the underlying tree network of a tree machine can be adapted to its corresponding compressed tree machine without loss of efficiency. We also showed that a compressed tree machine can be laied out on *VLSI* chips in a way that is more area-efficient than the H-tree layouts of tree machines. Our results provide a trade-off between different objectives, such as system cost and performance, in designing tree-based parallel computer architectures. We believe that similar trade-offs of cost effectiveness can be sought for many other existing networks. Our compressed trees can also be used to improve the cost-effectiveness of other architectures based on trees, such as mesh-of-trees.

For complete binary trees, there exists area-efficient layouts in which all leaf nodes are located on the boundary of the rectangle containing the layouts (e.g.[5]). One interesting problem is to find such layouts for compressed trees. Another aspect that may need further investigation is the embedding of compressed trees in other networks. There may exist useful networks for which embeddings of complete binary trees are not as good as the embeddings of compressed trees, and in such cases, it may be more desirable to simulate a compressed tree machine other than a tree machine.

IV. References

[1] Akl, S.G., *Parallel Sorting Algorithms,* Academic, Orlando, FL, 1985.

[2] Akl, S.G., *The Design and Analysis of Parallel Algorithms,* Prentice-Hall, Eaglewood Cliffs, NJ, 1989.

[3] Browning, S.A., "The Tree Machine: A Highly Concurrent Computing Environment", Ph.D. Thesis, Dept. of Computer Science, CIT, Pasadena, CA, 1980.

[4] Bently, J.L. and H.-T. Kung, "A Tree Machine for Searching Problems", Proc. Int. Conf. Parallel Processing, IEEE, pp.257-266, 1979.

[5] Brent, R.P. and H.-T. Kung, "On the Area of Binary Tree Layouts", *Information Processing Letters*, 11:1, pp.44-46, 1980.

[6] Bhatt, S.N. and C.E. Leiserson, "Minimizing the Longest Edge in a VLSI Layout", unpublished memorandum, Laboratory for Computer Science, MIT, Cambridge, Mass., 1981.

[7] Bhatt, S.N. and C.E. Leiserson, "How to Assemble Tree Machines", Proc. 14th Annual ACM Symposium on Theory of Computing, pp.77-84, 1982.

[8] Dekel, E. and S. Sahni, "Binary Trees and Parallel Scheduling", *IEEE Trans. Comput.*, C-32, pp.307-315, 1983.

[9] Fischer, M.J. and M.S. Paterson, "Optimal Tree Layout", Proc. 12th Annual ACM Symposium on Theory of Computing, pp.177-189, 1980.

[10] Horowize, E. and A. Zorat, "Divide-And-Conquer for Parallel Processing", *IEEE Trans. Comput.*, C-32, pp.582-585, 1983.

[11] Lipton, R.J. and J. Valdes, "Census Functions: A Approach to VLSI Upper Bounds", Proc, 21st Annual IEEE symposium on Foundations of Computer Science, pp.13-22, 1981.

[12] Mead, C. and L. Conway, *Introduction to VLSI Systems,* Addison-Wesley, Reading, Mass., 1980.

[13] Paterson, M.S., W.L. Ruzzo and L. Snyder, "Bounds on Minmax Edge Length for Complete Binary Trees", Proc. 13th Annual ACM Symposium on Theory of Computing, pp.293-299, 1981.

[14] Ruzzo, W.L. and L. Snyder, "Minimum Edge Length Planar Embeddings of Trees", in *VLSI Systems and Computations* (Kung, Sproull and Steele. eds.), pp.119-123, 1981.

[15] Song, S.W., "A Highly Concurrent Tree Machine for Database Applications", Proc. Int. Conf. Parallel Processing, IEEE, pp.259-268, 1980.

[16] Stout, Q.F., "Sorting, Merging, Selecting and Filtering on Tree and Pyramid Machines", Proc. Int. Conf. Parallel Processing, IEEE, pp.214-221, 1983.

[17] Thompson, C.D., "Area-Time Complexity for VLSI", Proc. 11th Annual ACM Symposium on Theory of Computing, pp.81-88, 1979.

[18] Thompson, C.D., "A Complexity Theory for VLSI", Ph.D. Thesis, Carnegie-Mellon Univ., Pittsburgh, Pa., 1981. Area-Time Complexity for VLSI",

[19] Ullman, J.D., *Computational Aspects of VLSI,* Computer Science Press, Rockville, MD, 1984.

[20] Valiant, L.G., "Universal Considerations in VLSI Circuits", *IEEE Trans. Comput.*, C-30, pp.135-140, 1981.

[21] Yao, A.C., "The Entropic Limitations of VLSI Computation" Proc. 13th Annual ACM Symposium on Theory of Computing, pp.308-311, 1981.

DYNAMIC PROGRAMMING ON LINEAR SYSTOLIC ARRAYS

Jean Frédéric MYOUPO
Laboratoire de Recherche en Informatique, C.N.R.S. UA 410, Bat. 490
Université Paris-Sud
91405 Orsay, France

Abstract: In this paper, we develop a variant of Ramakrishnan and Varman's linear algorithm [8] for dynamic programming problems that requires n cells and n^2+2n-1 time steps for its execution. Which is better than the $2n^2+2n-2$ time steps required by the execution of the previous algorithm.

Key Words: Design of Algorithms, Parallel Algorithms, Linear Systolic Array, Complexity, Dynamic Programming.

1- Introduction

Due to VLSI technology, hardware implementation of algorithmicly specialized parallel processors has become feasible and desirable. Recently, systolic arrays have received much attention. This is due to their simple control and timing, local interconnection between processors and modular designs. In pratice, linear arrays are more attractive than two-dimensional arrays. This is due to the fact that they have bounded I/O requirements [5]. In wafer containing faulty cells, a large percentage of non-faulty cells can be efficiently reconfigured into a linear array [6]. Synchronization between cells in a linear array can be achieved by a simple global clock whose rate is independent of the size of the array [2].There have been many efforts to formalize the synthesis of systolic arrays from algorithm descriptions, but only very regular algorithms in the form of systems of uniform recursive equations can be successfully treated (In a typical application, such arrays would be attached as a peripheral device to a host computer which would insert input values and extract ouput values from them). However, in pratical application there are certain classes of algorithms which are slightly irregular.

Dynamic Programming is one of the several widely used problem-solving techniques in Computer Science and Operation Research [1]. Parallel algorithms for this problem have been studied in [1, 3, 4, 7, 8, 9]. A linear array for the dynamic programming problems appeared in [8]. It uses n cells and requires $2n^2+2n-2$ time steps for its execution. Three of the channels of this array have their time delays (between two consecutive cells) depending on n: That is 2n+2, 2n+3 and 2n+4 time steps respectively.

In this paper, we derive a faster linear array algorithm from [8] using a geometric approach. It requires n cells and n^2+2n-1 time steps for its execution, which is faster than the one in [8]. Furthermore, only two of its channels have their time delays (between two consecutive cells) as functions of n: That is n+1 and n+3 time steps respectively.

The rest of this paper is organized as follows: In section 2, we set the dynamic programming problem and describe the linear array model. Next, in section 3, we give a proof of correctness of the algorithm. The conclusion follows in section 4.

2-Dynamic Programming and Linear array Model

The dynamic programming problem is to solve the following recurrence:

Given w(i, j), $1 \le i < j \le n$ and c(i, i+1), $1 \le i < n$, compute

$$c(i, j)=w(i, j)+\min_{i<k<j}(c(i, k)+c(k, j)) \text{ for } 1 \le i < j \le n.$$

This scheme can be used to solve many combinatorial optimization problems. Its essential feature is that the optimal solution to a problem is written as a recurrence relation in optimal solutions of subproblems. In this paper , we will restrict our attention on the construction of an optimal binary search tree which is a well-known example of dynamic programming:

$$c(i,j)=w(i, j)+\min_{i<k<j}(c(i, k)+c(k, j)), \ 1 \le i < j \le n+1. \qquad (1)$$

We compute the recurrence (1) on a linear array of n cells. Each cell has a local memory of size n+1. Our array contains four data channels, H_f, H_s, V_f and V_s, two controls channel, H_c and V_c and an address channel A_d as shown in Figure1.

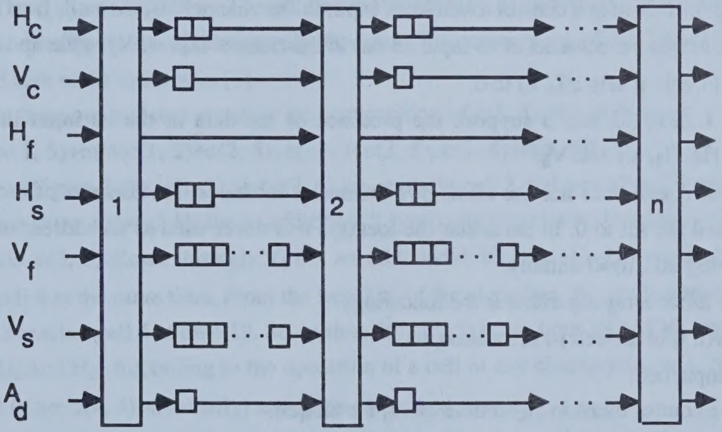


Figure 1.

Tokens travelling in the fast channel H_f are transfered in the slow channel H_s at some time; the same phenomenon happens to the tokens travelling in V_f, they are transfered in V_s at some time. The tokens travelling in H_c, V_c and A_d stay in the same channels as they traverse the array.

The tokens travelling in H_f, H_s, V_f, V_s, H_c, V_c and A_d encounter a delay (between two consecutive cells) of 1, 3, n+1, n+3, 3, 2, and 2 clock cycles respectively.

Let c(i, j) be the contents of the location s=n-i of the cell L=j-i, that is initialized to w(i, j); then

$$c(i, j)=\min(H_f(L)+V_s(L), V_f(L)+H_s(L)) \quad (2)$$

or

$$c(i, j)=\min(c(i, j), H_f(L)+V_s(L), V_f(L)+H_s(L)) \quad (3)$$

where $H_f(L)$, $V_f(L)$, $H_s(L)$ and $V_s(L)$ are the cell's inputs in H_f, V_f, H_s and V_s respectively. Let z(L, n) be the contents of the location n in the cell L. The operation of a cell L ($1 \leq L \leq n$) in any clock cycle is the following:

Step 1. If there is a control token in its input in the channel V_c, then it changes the contents of its input tokens on channels H_s and V_s to that of its input tokens on H_f and V_f respectively.

Step 2. If z(L, n)=0, then z(L, n)=c(i, j) and (2) is performed.

Step 3. If there are simultaneously a control token in its input in the channel V_c and a control token in its input in the channel H_c then it skips the next steps.

Step 4. If z(L, n)≠0 then (3) is performed.

Step 5. If there is a control token in its input in the channel H_c, then c(i, j)=z(L, n)+c(i, j), and it changes the contents of its input tokens in the channel H_f and V_f to the updated value of c(i, j). Finally it sets z(L, n) to 0.

Remark 1. Steps 2 and 3 suppose the presence of the data in the its input in the data channels H_f, H_s, V_f and V_s.

Remark 2. A cell does not use all its n+1 memories locations; the contents of those which are not used are set to 0. In particular the location n is never used as an address of a c(i, j); consequently z(L, n)=0 initially.

The linear array algorithm is the following:

1. Store w(i, j) in the cell j-i at location n-i.

2. At the input cell:

a). Insert a control token in H_c at time qn+2, for all $q \geq 0$.

b). Insert a control token in V_c at time qn+1, for all $q \geq 0$.

c). Insert an address token initialized to address s at time sn+2+p, for all p such that $-1 \leq p \leq n-2$.

This completes the description of the algorithm. As in [8] its effect is the following: Let s=n-i and y=j-i. Let c(i, j) be the token in the location s of the cell y. c(i, j) is computed and is ready in cell y at time sn+2+3(y-1). The cell y then starts transmitting c(i, j) in both H_f and V_f. c(i, j) travels in H_f for an additional y clock cycles and is transfered into H_s at cell 2y. It remains in H_s till eternity. Analogously, c(i, j) travels in V_f for an additional (n+1)y clock cycles before being transfered into V_s at cell 2y. It remains in V_s till eternity.

Proposition: The time of execution of the algorithm for a problem of dynamic programming of size n is n^2+2n-1.

Proof. c(1, n+1) is the last element to be computed in cell n. Its computation is completed at time $(n-1)n+2+3(n-1)=n^2+2n-1$.

Remark 3. We have obtained this array from geometric considerations. We consider the plane whose axes are cell's axis and time axis. Consider the embedding of the array in [8] in this plane. Then the trajectory of a token travelling through a channel with a time delay r (between two consecutive cells) is a line of slope r. From this embedding we have derived our algorithm (see Figures 2-(a) and 2-(b) for the geometric interpretation of the two linear arrays for n=4). The geometric interpretation of our algorithm is the following: c(i, j) has 2 paths which lead it to cell 2y: the first is a line of slope 1 and the second is a line of slope n+1. From this cell (numbered 2y), the first path changes to a line of slope 3 and the second changes to a line of slope n+3. The method consists of drawing the different trajectories of the elements c(i, j) in the plane (whose axes are cell's axis and time axis). At the intersection of these lines, (2) or (3) is performed. In figures 2-(a) and 2-(b), we have drawn the different paths of tokens travelling in H_c and V_c. We have not drawn all the trajectories of the elements c(i, j) (dotted lines), because they could make these figures unreadable. However one can easily verify on them that for n=4, the elements c(i, j) are correctly computed and that the linear array designed here is more efficient. These 2 linear algorithms ([8] and this paper) share the same geometric configuration. Consequently the proof of correctness of the linear algorithm presented here is the same as in [8].

Example: Suppose n=4 and consider the computation of c(1, 5). We know that
c(1, 5)=w(1, 5)+min(c(1, 2)+c(2, 5), c(1, 3)+c(3, 5), c(1, 4)+c(4, 5)).

From Figure 2-(b), c(1, 3) and c(3, 5) are ready in cell 2 at times 17 and 9 respectively. c(1, 3) then flows through H_f for an additional 2 (y=2) clock cycles and reaches cell 4 (2y=4) at time 19. c(3, 5) flows through V_f for an additional 10 (2(n+1)=10) clock cycles and reaches cell 4 at the same time. From the step 2.b) of the algorithm, the control token inserted at time 13 reaches cell 4 at time 19. So, at time 19, c(3, 5) is on both V_f and V_s and c(1, 3) is on both H_f and H_s. According to the operation of a cell in any clock cycle, w(1, 5) is stored in z(4, 4) (since z(4, 4)=0 initially) and (2) is performed; that is

c(1, 5)=min(c(1, 3)+c(3, 5), c(1, 3)+c(3, 5)).

From Figure 2-(b) again, c(2, 5) is ready at time 16 in cell 3. It travels on V_f from cell 3 and arrives in cell 4 at time 21. c(1, 2) is ready in cell 1 at time 14. It then travels for an additional 1 clock cycle on H_f till it reaches cell 2 (y=1) at time 15. It is then transfered into H_s. It travels in H_s for an additional 6 clock cycles till it reaches cell 4 at time 21. So at time 21, c(1, 2) and c(2, 5) arrive in H_s and V_f respectively in cell 4. Similary it can be verified that c(1, 4) and c(4, 5) also arrive in cell 4 in H_f and V_s respectively at time 21. Since w(1,5) has been stored in z(4, 4), then z(4, 4)≠0 and (3) is performed; that is

c(1, 5)=min(c(1, 5), c(1, 2)+c(2, 5), c(1, 4)+c(4, 5)).

At time 23, the control token travelling in H_c (inserted at time 14) and the address of c(1, 5) (inserted at time 17) arrive in cell 4. Then c(1, 5)=c(1, 5)+z(4, 4).

3. Proof of Correctness

The approach here is geometric. Note that through this section, the origin of the plane is (0, 0) instead of (1, 1) as supposed previously in this paper. We consider the plane defined above and the trajectories of the different elements travelling in the array. Our proof is along similar lines to the proof in [8]:

Lemma: Let y=j-i and s=n-i, $1 \le i < j \le n+1$.

1)- c(I, j) is transfered into H_f and V_f in cell y at time sn+1+3(y-1).

2)- a) c(i, j) travels in H_f for y clock cycles and is then transfered into H_s in cell 2y.

b) c(i, j) travels in V_f for (n+1)y clock cycles and is then transfered into V_s in cell 2y.

Proof. 1)- c(i,j) will be transfered into H_f and V_f only if there is a control token present in H_c at the input of cell y at time sn+1+3(y-1). This means that this control token must have been inserted into H_c of cell 1 at time sn+1+3(y-1)-3(y-1) (the time delay between 2 consecutive cells through H_c is 3) i.e sn+1. By a) of the algorithm, a control token is inserted into the array through H_c at time qn+1, for all $q \ge 0$.

2)- We will prove a); the proof of b) is similar.

a) By 1) of the lemma, c(i, j) is transfered into H_f at time t1=sn+1+3(y-1). In y additional clock cycles, it will reach cell 2y. In order for c(i, j) to be transfered into H_s at cell 2y, it must meet a control token in V_c at the input of cell 2y at time t1+y. This means that this control token must have been inserted into the array at time t2=t1+y-2(2y-1) (the time delay through V_c is 2). Substituting s=n-i and y=j-i, t2 reduces to sn. From b) of the algorithm, a control token is inserted into the array through V_c at time qn, for all $q \ge 0$. Keep in mind that here, the origin of the plane is (0, 0) instead of (1, 1). The array is translated in such a way that c(n, n+1) is computed and is transfered on both H_f and V_f at time 1 instead of 2 as in Figures 2-(a) and 2-(b).

Theorem: $c(i, j)=w(i, j)+\min\limits_{i<k<j}(c(i, k)+c(k, j))$ when it is transfered into H_f and V_f.

Proof. We prove this as in [8], by induction on y=j-i.

Basis: y=1. The correct value of c(i, j) when y=1 is its initial value w(i, j) which is stored in location s of cell 1. At time sn+1, address s and a control token are inserted in A_d and H_c respectively. So w(i, j) gets transfered into H_f and V_f.

Inductive step: We have to show that the theorem holds for all i' and j' such that j'-i'=y+1. Let i'=i+a-1 and j'=a+j. We then have to prove that

$$c(i+a-1, a+j)=w(i+a-1, a+j)+\min\limits_{i+a-1<k<a+j}(c(i+a-1, k)+c(k, a+j)).$$

To prove this, we must show the following:

i)- c(i+a-1, k) and c(k, a+j) meet at cell y+1 before c(i+a-1, a+j) is transfered.

ii)- When they meet, the address in A_d at the input is n-i-a+1.

By the inductive hypothesis and part 1) of the lemma, $c(i+a-1, k)$ is correctly computed when it is transfered into H_f and V_f at cell $k-i-a+1$ at time $t1=(n-i-a+1)n+1+3(k-i-a)$. It then travels in H_f for an additional $(k-i-a+1)$ clock cycles. Subsequently, it travels in H_s till it reaches cell $y+1$. Let $t2$ be the time taken to reach cell $y+1$ after the transfer. Then $t2=(k-i-a+1)+3(y+1-2k+2i+2a-2)$, which can be simplified to $2i+3j+5a-5k-2$; then

$$t1+t2=(n-i-a+1)n-2k+2a-i+3j-1. \qquad (4)$$

By the inductive hypothesis again, $c(k, a+j)$ is correctly computed in cell $a+j-k$ when it is transfered into H_f and V_f at time $t3=(n-k)n+1+3(a+j-k-1)$. It then travels in V_f till it reaches cell $y+1$. Let $t4$ be this travel time. So $t4=(n+1)(y+1-a-j+k)$.

$t3+t4=(n-k)n+1+3(a+j-k-1)+(n-1)(y+1-a-j+k)=(n-i-a+1)n-2k+2a-i+3j-1$ which is the same as (4) since $y=j-i$. We next show that (4) is less than the time at which $c(i+a-1, a+j)$ is transfered into H_f and V_f. By 1) of the above lemma, this time is $(n-i-a+1)n+1+3(j-i)$. We then have to show that

$$(n-i-a+1)n-2k+2a-i+3j-1<(n-i-a+1)n+1+3(j-i).$$

Which reduces to showing that

$$-2k+2a-i+3j<2+3j-3i.$$

This last inequality reduces to $k>i+a-1$. Which is true by the definition of k.

In the precedent proof, we had assumed that $c(i+a-1, k)$ travels in V_f alone. We can also show in the symmetric case where $c(i+a-1, k)$ travels in H_f and $c(k, a+j)$ travels in V_f followed by V_s that they still meet at cell $y+1$.

Finally, we must show that the address at the input channel A_d of the cell $y+1$ is $n-i-a+1$. This address must have been inserted in the address channel A_d of cell 1 at time (4)-2(j-i); this time can be simplified to $(n-i-a+1)n-2k+2a+j+i-1$. From section c) of the algorithm, this address is $n-i-a+1$ if

$$-1\leq-2k+2a+i+j-1\leq n-2 \qquad (5)$$

By the definition of k, we have $k<a+j$; then $2k<2a+2j$. Now $i+j<2j$ i.e $i+j+1\leq 2j$, then $2k<2a+j+i+1$, which implies that $-2k+2a+i+j-1>-2$ or $-2k+2a+i+j-1\geq-1$; which is the first inequality in the expression (5).

Now $k<a+j\leq n$ and $i+a-1<k$ by the definition of k. Then $i+a-1-k<0$ and $i+a-1-k+a+j<n$, i.e $-k+2a+i+j-1\leq n-1$. Sincek>0, $-2k+2a+i+j-1\leq n-2$, which proves the second inequality of (5).

4. Conclusion

In this paper, we have presented a linear array algorithm for dynamic programming; we have used a geometric approach to design this array. Its characteristics make it more easy to handle than the one in [8] (see TABLE I for the comparison of the 2 arrays). It runs in n^2+2n-1time steps instead of $2n^2+2n-2$ as in [8]. This work was motivated by the fact that we want to design modular linear systolic arrays for dynamic programming problem. The array of this paper is just a step to our goal. However it would be interesting to design at least a linear array in which the time delay through each channel (between 2 consecutive cells) does not depend on n.

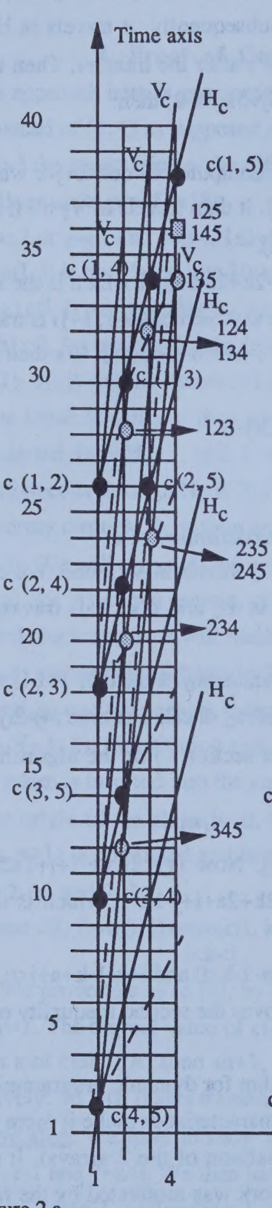
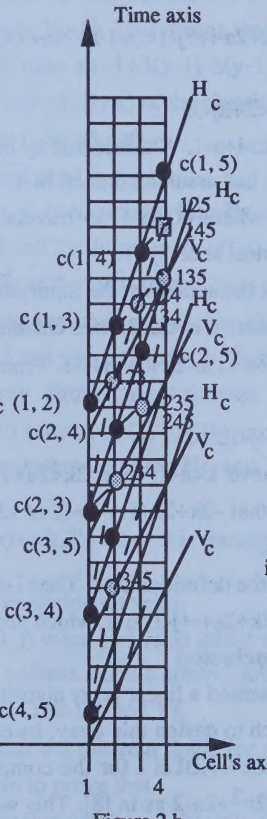

Time axis
V_c
H_c
40
c(1, 5)
125
145
35
c (1, 4)
135
124
134
30
c (1, 3)
123
c (1, 2)
25
c(2, 5)
235
245
c (2, 4)
20
234
c (2, 3)
15
c (3, 5)
345
10
c(3, 4)
5
c(4, 5)
1
1
4

Figure 2.a.
Geometric interpretation of
the Algorithm in [8] for n=4.


Time axis
H_c
c(1, 5)
125
145
c(1, 4)
V_c
135
124
c (1, 3)
134
c(2, 5)
c (1, 2)
235
245
c(2, 4)
234
c(2, 3)
c(3, 5)
345
c(3, 4)
c(4, 5)
Cell's axis
1
4

● Point at which the computation point c(i, j) is completed and is transfered in both H_f and V_f.

▨ Partial computation point of c(i, j) using the expression (2).

▨ Partial computation point of c(i, j) using the expression (3).

ijk represents the expression c(i, k)+c(k, j) involved in the computation of c(i, j).

Figure 2.b.
Geometric interpretation of the
Algorithm of this paper for n=4.

TABLE I

Comparison of Linear Algorithms for Dynamic Programming

Characteristics of the array		This paper	Method in [8]	Method in [9]
Number of Cells		n^2+1	n	n
Time of Execution		$6n^2-n-3$	n^2+2n-1	$2n^2+2n-2$
Time Delay between two Consecutive Cells through the Channel...	V_f	6	n+1	2n+2
	V_s	8	n+3	2n+4
	V_c	2	2	2n+3

References

[1] K. Q. Brown: "Dynamic Programming in Computer Science", CMU Tech. Report (February 1979).

[2] A. L. Fischer and H. T. Kung:"Synchronizing Large VLSI Processor Arrays", *Proc. Tenth Annual IEEE/ACM Symposium on Computer Architecture,* June 1983, PP. 54-58.

[3] L. J. Guibas, H. T. Kung and C. D. Thompson:"Direct VLSI Implementation of Combinatorial Algorithms", *Proc. Conf. very Large Scale Integration: Architecture, Design, Fabrication,* California Institite of Technology (Jauary 1979), PP. 509-525.

[4] D. E. Knuth:" *The Art of Computer Programming*", Vol. 3 Sorting and Searching, Addison-Wesley (1973).

[5] H. T. Kung: "Why Systolic Architecture?", *IEEE Computer* 15 (1) January, 1980, PP. 37-46.

[6] F. T. Leighton and C. E. Leiserson:"Wafer-Scale Integration of Systolic Arrays", *Proc. Twenty third Symp. Foundations of Computer Science,* November 1982, PP. 297-311.

[7] B. Louka and M. Tchuente:"Dynamic Programming on Two-Dimensional Systolic Arrays", *Infom. Process. Lett.* 29, (1988), PP. 97-104.

[8] I. V. Ramakrishnan and P. J. Varman:" Dynamic Programming and Transitive Closure on Linear Pipelines", *Proc. Int. Conf. Parallel Processing*, St. Charles, August 1984.

[9] F. L. Van Scoy:"The parallel Recognition of Classes of Graphs", *IEEE-TC, Vol. C-29, No. 7* (July 1980), PP. 563-570.

A Heuristic of Scheduling Parallel Tasks and Its Analysis[1]

Qingzhou Wang and Kam Hoi Cheng
Computer Science Department
University of Houston
Houston, Texas 77204-3475

Abstract

In this paper, we investigate the problem of non-reconfigurable non-preemptive scheduling of parallel tasks in a homogeneous system of processors. Given a task precedence graph where each task T may be executed on up to $\delta(T)$ processors, the problem is to find a schedule such that the makespan is minimized. A scheduling policy is non-reconfigurable if when more processors become available after a task T has been scheduled on p processors, $p < \delta(T)$, they cannot join the execution of task T. It is non-preemptive if once a task has started execution, it is executed without interruption until completion. The problem of scheduling parallel tasks is NP-hard. The performance ratio of the makespan produced by the original list scheduling algorithm proposed by Graham [4] to the optimal makespan is shown in [6] to be bounded by $\Delta + \frac{m-\Delta}{m}$, where m and Δ are the number of processors in the system and the maximum degree of parallelism in any task, respectively. To provide a performance improvement, we propose to use the *Earliest Completion Time* (ECT) algorithm and show that its performance ratio is bounded by $3 - \frac{2}{m}$.

1. Introduction

Let $\mathcal{P} = \{P_i : i = 1, \ldots, m\}$ be m identical processors, and $\mathcal{T} = \{T_j : j = 1, \ldots, n\}$ be n tasks. For two tasks $T_j, T_{j'} \in \mathcal{T}$, $T_j \prec T_{j'}$ means that the task $T_{j'}$ can not be started until the task T_j has been completed. For each task $T_j \in \mathcal{T}$, it may be scheduled to run on p_j processors where $1 \leq p_j \leq \delta_j$, and δ_j is T_j's maximum degree of parallelism. If a task T_j is scheduled to run on p_j processors, then p_j is called the *scheduled parallelism* of

---

[1] This research is based in part upon work supported by the Texas Advanced Research Program under grant 1028-ARP.

T_j. For each task T_j, τ_j defines the size of the task and if its scheduled parallelism is p_j, the execution time required will be τ_j/p_j.

In this paper, we are concerned with *non-reconfigurable non-preemptive* scheduling policies. This is based on the postulation that altering the degree of parallelism of a task during its execution or task preemption requires substantial overhead. For any schedule S, there are *starting* time $b_S(T_j)$, *completion* time $c_S(T_j)$, and scheduled parallelism $p_S(T_j)$, associated with every task $T_j \in \mathcal{T}$. A schedule is *feasible* if the scheduled parallelism of each task is no greater than its maximum degree of parallelism, it requires no more than m processors at any moment, and it preserves the precedence relationships among tasks, i.e.,

$$\begin{aligned} &1 \le p_S(T_j) \le \delta_j, && \forall T_j \in \mathcal{T}; \\ &\sum_{\substack{1 \le j \le n \\ b_S(T_j) \le t < c_S(T_j)}} p_S(T_j) \le m, && \forall t \in [0, \infty); \\ &T_{j'} \prec T_{j''} \Rightarrow c_S(T_{j'}) \le b_S(T_{j''}), && \forall T_{j'}, T_{j''} \in \mathcal{T}. \end{aligned} \tag{1.1}$$

In a feasible schedule S, the ready time of a task T_j, $r_S(T_j)$, is defined to be the latest completion time of all T_j's predecessors, i.e., $r_S(T_j) = \max_{T_{j'} \prec T_j} c_S(T_{j'})$. If a task T_j does not have any predecessor in $\mathcal{T}$, then $r_S(T_j) = 0$. The performance of a feasible schedule S is measured by its makespan, $M_S(\mathcal{T}, \mathcal{P}) = \max_{T_j \in \mathcal{T}} c_S(T_j)$. The objective of the scheduling problem is to find a feasible schedule such that the makespan is minimized. Let $M_{OPT}(\mathcal{T}, \mathcal{P})$ be the makespan of an optimal schedule.

The problem of task scheduling has been studied extensively, and has many variations [5]. The introduction of parallel tasks into scheduling problems can be found in [1]-[3]. Recently, Belkhale and Banerjee [1] has worked on the problem where all tasks are independent. The justification of defining the maximum degree of parallelism for each task is that in many operations, complete parallelism is difficult to achieve, but limited parallelism may be accomplished easily. At the same time, a linear reduction in execution time is a reasonable assumption if only simple parallelizations were attempted on tasks. Our problem is NP-hard since the NP-hard problem of non-preemptive scheduling of sequential tasks [4] is a special case of our problem when the degree of parallelism is limited to 1 for every task. Since polynomial time optimal solution is unlikely for any NP-hard problem, we are interested in polynomial time approximation algorithms. A heuristic scheduling algorithm, H, has a performance ratio α if for all problem instances, it can guarantee that

$$\frac{M_H(\mathcal{T},\mathcal{P})}{M_{OPT}(\mathcal{T},\mathcal{P})} \leq \alpha \tag{1.2}$$

where $M_{OPT}(\mathcal{T},\mathcal{P})$ and $M_H(\mathcal{T},\mathcal{P})$ are the optimal and the heuristic makespans respectively.

2. The Original List Scheduling Algorithm

The list scheduling algorithm, LS, proposed by Graham [4] for sequential tasks is formally presented in Algorithm 1. In the algorithm, t_{next} is basically the earliest time at which there is at least one free processor. Initially $t_{next} = 0$. At time t_{next}, the algorithm selects the first ready task from the given task list for execution. If there is no ready task at time t_{next}, t_{next} is then assigned the earliest time at which at least one more running task completes its execution. The algorithm repeats these simple steps until the task list becomes empty.

Applying algorithm LS to a list of parallel tasks, $\mathcal{T}$, is essentially giving each task a scheduled parallelism of 1, and it may cause under-utilization of the processor system. It has been shown in [6] that $\frac{M_{LS}(\mathcal{T},\mathcal{P})}{M_{OPT}(\mathcal{T},\mathcal{P})} \leq \Delta + \frac{m-\Delta}{m}$, where Δ is the maximum degree of parallelism in any task, and the bound is tight for all Δ values. Note that when $\Delta = 1$, the bound is reduced to that obtained in [4] for scheduling sequential tasks.

```
        input(task list $\mathcal{T}$)
        $t_{next} \leftarrow 0$
        repeat
            $t_{next} \leftarrow \max\{t_{next},$ time when at least one processor is free$\}$
            if (there exists a ready task in the list $\mathcal{T}$ at $t_{next}$) then
                remove the first ready task $T_j$ from the list $\mathcal{T}$
(†)             execute $T_j$ on processor $P_i$ where $P_i$ is free at $t_{next}$
            else
                $t_{next} \leftarrow$ time with at least one more task completes
            endif
        until $\mathcal{T} = \emptyset$
```

Algorithm 1. The Original List Scheduling Algorithm

3. The ECT Scheduling Algorithm

The ECT algorithm is basically a list scheduling algorithm except that the (†) statement in Algorithm 1 is replaced by a procedure to determine the execution time and the scheduled parallelism of a ready task. The idea is to complete the ready task as soon as possible, and it is called the *Earliest Completion Time* procedure (Algorithm 2). For each parallel task scheduled by the ECT algorithm, the time at which the task T_j is submitted to the ECT procedure, $a_S(T_j)$, is called its *preparation* time. Obviously,

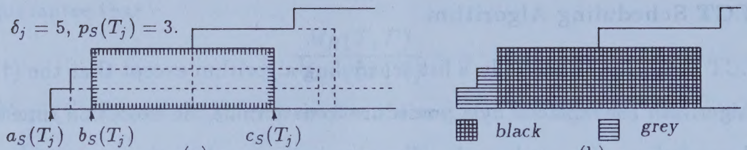
$$r_S(T_j) \leq a_S(T_j) \leq b_S(T_j) \leq c_S(T_j), \quad \forall T_j \in \mathcal{T}. \tag{3.1}$$

We say that the task T_j is being prepared from time $a_S(T_j)$ to $b_S(T_j)$. We will present a brief outline of the essential steps of the proof which shows that the performance ratio of the ECT algorithm is bounded by $3 - \frac{2}{m}$. Details of the proof may be found in [7].

```
procedure Earliest_Completion($t_{next}$, $T_j$)
$p' \leftarrow 1$; $t_{p'} \leftarrow t_{next}$; $t_{earlist} \leftarrow t_{next} + \tau_j$
for $p \leftarrow 2$ to $\delta_j$ do
    find the earliest time $t_p \geq t_{next}$ with $p$ free processors
    $\tilde{t}_p \leftarrow t_p + (\tau_j/p)$
    if $\tilde{t}_p < t_{earliest}$ then $t_{earliest} \leftarrow \tilde{t}_p$; $p' \leftarrow p$; $t_{p'} \leftarrow t_p$
enfor
schedule $T_j$ on $p'$ free processors at time $t_{p'}$; $t_{next} \leftarrow t_{p'}$
```

Algorithm 2. The Earliest Completion Time Procedure

To visualize processor allocation, for each processor $P_i \in \mathcal{P}$, we present its activities by using a half-open strip starting from time $t = 0$. The activities of all processors in $\mathcal{P}$ may be expressed by m stacked up strips, P_i, $i = 1, \ldots, m$. For each strip, P_i, it is colored *white* initially. Since all processors are identical, we may assume without loss of generality that at time t_p, the available processors are $P_1, \ldots, P_p$ where $p \in \{1, \ldots, \delta_j\}$. Figure 1(a) illustrates the fitting process by the ECT procedure for a task with $\delta_j = 5$ and $p' = 3$. For the strip (processor) P_p, $1 \leq p \leq \delta_j$, we color it *grey* from time t_p to the task's starting time $b_S(T_j)$ if $t_p < b_S(T_j)$. For the p' strips on which the task was scheduled, they are colored *black* from time $b_S(T_j) = t_{p'}$ to $t_{p'} + \tau_j/p'$. The coloring for



(a) (b)

Figure 1. Fitting and Coloring in Earliest Completion Time Procedure

the task in Figure 1(a) is a black rectangle with a staircase shaped grey area to its left (Figure 1(b)). Note that it is impossible to re-color any area that is already black or grey. It is easy to see that a necessary and sufficient condition for a section of a strip to remain white is when the corresponding processor is free but no ready task is available. An interval $[t, t')$ is incompletely colored if one or more strips in that interval remain white, $t' \leq M_{ECT}(\mathcal{T}, \mathcal{P})$; otherwise, it is completely colored. We can further assume that each such interval is of maximal length, i.e., it is not contained in another incompletely colored interval. Let k be the number of incompletely colored intervals. These intervals may be represented by

$$[t_1, t_2), \ldots, [t_{2k-1}, t_{2k}). \tag{3.2}$$

Lemma 1. In the ECT schedule S, $[r_S(T_j), b_S(T_j)) \cap \{\cup_{i=1}^{k} [t_{2i-1}, t_{2i})\} = \emptyset$ for any task T_j. ■

The next lemma establishes the fact that the incompletely colored intervals in any ECT schedule must be covered by a chain of tasks.

Lemma 2. (Covering Lemma) Let S be the ECT schedule and $[t_{2i-1}, t_{2i})$, $i = 1, \ldots, k$, its incompletely colored intervals. There must be a chain of tasks in $\mathcal{T}$,

$$T_{j_h} \prec \ldots \prec T_{j_i} \prec \ldots \prec T_{j_1}, \tag{3.3}$$

such that $\cup_{i=1}^{k} [t_{2i-1}, t_{2i}) \subseteq \cup_{i=1}^{h} [b_S(T_{j_i}), c_S(T_{j_i}))$. ■

To analyze the size of the black areas, we examine closely how tasks were prepared and scheduled. A task is called a *wide* task if its maximum degree of parallelism is at least $\lceil m/2 \rceil$, otherwise, it is called a *narrow* task. In other words, $\mathcal{T}$ is divided into the subsets of wide and narrow tasks, $\mathcal{T} = \mathcal{T}_W \cup \mathcal{T}_N$. Then, all remaining intervals not covered by the wide tasks are further divided into two groups of subintervals, H and I. First we will prove that in the intervals covered by the preparation and execution of wide tasks, the black area is at least half the total area. We will further show that at least half of the area in H is colored black, and I is covered by the shortest possible execution time of a subchain from the task chain of (3.3). Using these results, we can establish the $3 - \frac{2}{m}$ performance bound of the ECT algorithm.

Let T_j be a wide task. A wide task T_h is an *immediate follower* of T_j if it satisfies

$$c_S(T_h) = \max_{a_S(T_j) < a_S(T_{j'}) \leq c_S(T_j),\ T_{j'} \in \mathcal{T}_W} c_S(T_{j'}) \quad \text{and} \quad c_S(T_j) < c_S(T_h). \tag{3.4}$$

Similarly, a wide task T_i is a *distant follower* of T_j if it satisfies

$$c_S(T_i) = \max_{a_S(T_{j'}) = t,\ T_{j'} \in \mathcal{T}_W} c_S(T_{j'}) \quad \text{where} \quad t = \min_{c_S(T_j) < a_S(T_{j'}),\ T_{j'} \in \mathcal{T}_W} a_S(T_{j'}). \tag{3.5}$$

To construct sequences of wide tasks, the first wide task $T_{(1,1)}$ is selected to satisfy

$$c_S(T_{(1,1)}) = \max_{a_S(T_{j'}) = t,\ T_{j'} \in \mathcal{T}_W} c_S(T_{j'}) \quad \text{where} \quad t = \min_{0 \leq a_S(T_{j'}),\ T_{j'} \in \mathcal{T}_W} a_S(T_{j'}). \tag{3.6}$$

In general, suppose we have already found the wide task $T_{(i,j)}$. If $T_{(i,j)}$ has an immediate follower, we will designate that follower as $T_{(i,j+1)}$. If it does not have an immediate follower, but has a distant follower, we will designate the distant follower as $T_{(i+1,1)}$. The selection is repeated until we can find neither an immediate follower nor a distant follower. The wide tasks selected in this process are shown as follows:

$$\begin{array}{ccc} T_{(1,1)}, & \ldots, & T_{(1,j_1)} \\ \vdots & \cdots & \\ T_{(l,1)}, & \ldots, & T_{(l,j_l)}. \end{array} \tag{3.7}$$

Lemma 3. There is no wide task that is either prepared or executed in the following intervals: $[0, a_S(T_{(1,1)})), \ldots, [c_S(T_{(i,j_i)}), a_S(T_{(i+1,1)})), \ldots, [c_S(T_{(l,j_l)}), M_{ECT}(\mathcal{T}, \mathcal{P}))$. ∎

Next we divide the interval covered by a single row of wide tasks:

$$T_{(i,1)}, \ldots, T_{(i,j_i)}, \tag{3.8}$$

into subintervals: $[t_{h-1}, t_h)$, $h = 1, \ldots, j_i$, where $t_0 = a_S(T_{(i,1)})$, and $t_h = c_S(T_{(i,h)})$. Suppose we remove the black areas associated with the tasks of (3.8) in the ECT schedule, and rearrange the remaining black strips in each individual subintervals into staircase shaped regions in such a way that the size of the remaining black colored regions is unchanged (Figure 2). Within the subinterval $[t_{h-1}, t_h)$, we now try to fit a largest

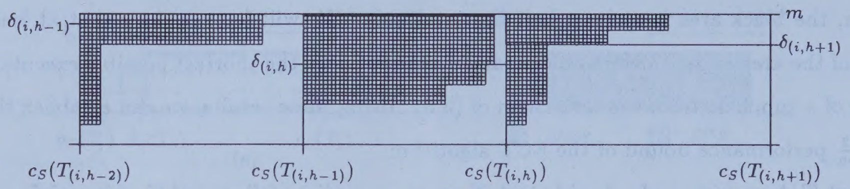


Figure 2. Black Colored Regions After Rearrangement

possible black rectangle with height not exceeding $\delta_{(i,h)}$ into the area at the lower right hand corner, and Figure 3 shows such a fitting for some subintervals.

Lemma 4. For each subinterval $[t_{h-1}, t_h)$, $1 \leq h \leq j_i$, the area of the largest black rectangle fitted will not exceed $\tau_{(i,h)}$.

Proof: If this is not true, we can schedule the task $T_{(i,h)}$ differently and achieve a completion time earlier than $t_h = c_S(T_{(i,h)})$. This is in contradiction with the scheduling criterion used in the ECT procedure. ∎

Now we estimate the total size of all black areas in each subinterval $[t_{h-1}, t_h)$. Let R_h and S_h be the size of the rectangle at the lower right hand corner and the staircase shaped region at the upper left hand corner, respectively. Furthermore, let L be the length of the interval, i.e., $L = t_h - t_{h-1}$, and δ the maximum number of strips (processors) that task $T_{(i,h)}$ can possibly use, i.e., $\delta = \delta_{(i,h)} = \delta(T_{(i,h)})$. To make the mathematical analysis simpler, we analyze the mirror reflection of the region (Fig. 4(a)).

Lemma 5. For any positive integer $\delta \geq \lceil m/2 \rceil$, $R_h + S_h \geq m \cdot L/2$.

Proof: The basic idea of the proof is to use an estimation of $R_h + S'_h$ in Figure 4(b). Since R_h is the largest possible rectangle that can be fitted, we must have $S_h > S'_h$, and hence $R_h + S_h > R_h + S'_h$. Mathematical analysis shows that the area of $R_h + S'_h$ is at least $m \cdot L/2$. ∎

By Lemmas 4 and 5, we can conclude that the total black area occurred between t_0 and t_{j_i} is at least $m(t_{j_i} - t_0)/2$. Let W_i and W_i^b be the available and the utilized processing power in the i^{th} wide task sequence in (3.7), respectively. The following inequality is obvious by applying the result to all l wide task sequences,

$$\sum_{i=1}^{l} W_i^b \geq \frac{1}{2} \sum_{i=1}^{l} W_i. \tag{3.9}$$

Now we investigate the intervals that were not covered by the preparation and the

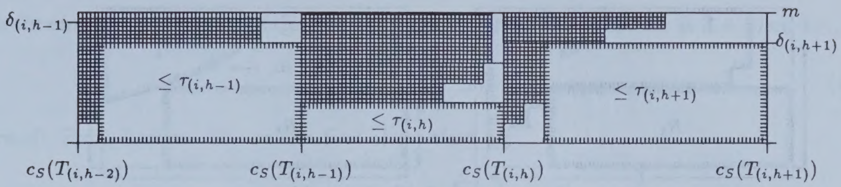


Figure 3. Fitting of Largest Rectangles

execution of wide tasks' sequences in (3.8). For W_i, $i = 1, \ldots, l$, let $[t^{(2i-1)}, t^{(2i)})$ be its corresponding interval on the x-axis. We further use W to denote the union of all such intervals, i.e., $W = \cup_{i=1}^{l} [t^{(2i-1)}, t^{(2i)})$. Excluding the intervals in W, we have $l+1$ intervals that are not covered by the l sequences of wide tasks:

$$[t^{(0)}, t^{(1)}), [t^{(2)}, t^{(3)}), \ldots, [t^{(2i)}, t^{(2i+1)}), \ldots, [t^{(2l)}, t^{(2l+1)}), \tag{3.10}$$

where $t^{(0)} = 0$ and $t^{(2l+1)} = M_{ECT}(\mathcal{T}, \mathcal{P})$.

Lemma 6. In an ECT schedule, we can not have grey and white strips coexist at any time $t \in [0, M_{ECT}(\mathcal{T}, \mathcal{P}))$. ■

We further subdivide the intervals in (??) into *more than half colored* and *at most half colored* categories. An interval is more than half colored if at all time in the interval, the number of white strips $< m/2$. On the other hand, in an at most half colored interval, the number of white strips $\geq m/2$ at all time t. For each time interval $[t^{(2i)}, t^{(2i+1)})$, $i = 0, \ldots, l$, let

$$[t_1^{(2i)}, t_2^{(2i)}), \ldots, [t_{2j-1}^{(2i)}, t_{2j}^{(2i)}), \ldots, [t_{2k_i-1}^{(2i)}, t_{2k_i}^{(2i)}), \tag{3.11}$$

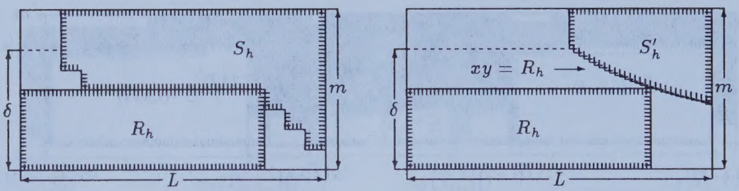
be its more than half colored intervals. We use H to denote the union of all such subintervals, i.e., $H = \cup_{i=0}^{l} \cup_{j=1}^{k_i} [t_{2j-1}^{(2i)}, t_{2j}^{(2i)})$.

Lemma 7. For any more than half colored interval $[t_{2j-1}^{(2i)}, t_{2j}^{(2i)})$, the rectangle $m \times (t_{2j}^{(2i)} - t_{2j-1}^{(2i)})$ has more than half of its area colored black.

Proof: Based on Lemma 6, we can consider the following two cases.

1. All strips in $[t_{2j-1}^{(2i)}, t_{2j}^{(2i)})$ are colored black and white: the statement in the theorem is true based directly on the definition of the more than half colored interval.

2. All strips in $[t_{2j-1}^{(2i)}, t_{2j}^{(2i)})$ are colored black and grey: we can conclude from Lemma 3 that the grey area in $[t_{2j-1}^{(2i)}, t_{2j}^{(2i)})$ must be introduced during the preparation of a narrow task. At any moment of preparing a narrow task in the ECT procedure, less than $\lfloor m/2 \rfloor$



(a) (b)

Figure 4. Estimating the Black Colored Areas

strips are colored grey. Therefore, more than half of the total rectangular area is colored black. ■

Lemma 8. If a task T_j is scheduled to start execution in an incompletely colored interval $[t', t'')$, the task T_j must be scheduled to run at its maximum degree of parallelism, i.e.,

$$b_S(T_j) \in [t', t'') \Rightarrow p_S(T_j) = \delta_j. \quad ■$$

Lemma 9. For an at most half colored interval $[t_{2j}^{(2i)}, t_{2j+1}^{(2i)})$ and a narrow task T_k,

$$t_{2j}^{(2i)} \in [b_S(T_k), c_S(T_k)) \Rightarrow c_S(T_k) - t_{2j}^{(2i)} \le \frac{\tau_k}{\delta_k}. \quad ■$$

Let I be the union of all at most half colored intervals that are not yet covered by the wide tasks, i.e.,

$$I = [0, M_{ECT}(\mathcal{T}, \mathcal{P})) - W - H. \tag{3.12}$$

From the task chain T_{j_i}, $i = 1, \dots, h$, constructed in Lemma 2, there is a subchain of narrow tasks

$$T_{j'_g} \prec \dots \prec T_{j'_i} \prec \dots \prec T_{j'_1}. \tag{3.13}$$

This chain of narrow tasks not only covers the intervals in I, it also provides an upper bound on the total length of these intervals. Lemma 10 shows that the total length of intervals in I does not exceed the execution time of the narrow task chain (??) even when it is executed with maximum parallelism.

Lemma 10. $\sum_{i=1}^{g} \tau_{j'_i}/\delta_{j'_i} \ge |I|$, where $|I|$ is the total length of intervals in I. ■

Let us use $H_{(i,j)}$ to denote the rectangle of size $m \times (t_{2j}^{(2i)} - t_{2j-1}^{(2i)})$, and whose bottom edge is the interval $[t_{2j-1}^{(2i)}, t_{2j}^{(2i)})$. According to Lemma 7, at least one half of the area in $H_{(i,j)}$ is black. In other words, if $H_{(i,j)}^{b}$ denotes the black area in $H_{(i,j)}$, we must have

$$\sum_{i=0}^{l} \sum_{j=1}^{k_i} H_{(i,j)}^{b} \ge \frac{1}{2} \sum_{i=0}^{l} \sum_{j=1}^{k_i} H_{(i,j)}. \tag{3.14}$$

Theorem 1. The overall performance bound of the ECT schedule is $3 - 2/m$, i.e.,

$$\frac{M_{ECT}(\mathcal{T},\mathcal{P})}{M_{OPT}(\mathcal{T},\mathcal{P})} \le 3 - \frac{2}{m}. \tag{3.15}$$

Proof: From Lemma 10, we can first show that

$$M_{OPT}(\mathcal{T},\mathcal{P}) \ge \sum_{i=1}^{g} \frac{\tau_{j_i'}}{\delta_{j_i'}} \ge |I|.$$

Let $|H|$ and $|W|$ denote the total length of more than half colored intervals in (??) and intervals covered by wide tasks in (3.7), respectively. By calculating the total black areas, we can next show that

$$M_{OPT}(\mathcal{T},\mathcal{P}) \ge \frac{1}{m}|I| + \frac{1}{2m}\left(\sum_{i=0}^{l}\sum_{j=1}^{k_i} H_{(i,j)} + \sum_{i=1}^{l} W_i\right) = \frac{1}{m}|I| + \frac{1}{2}(|H| + |W|).$$

After combining the above two inequalities, we have

$$M_{ECT}(\mathcal{T},\mathcal{P}) = |I| + |H| + |W| \le \ (3 - \tfrac{2}{m})M_{OPT}(\mathcal{T},\mathcal{P}). \quad \blacksquare$$

Corollary 1. [7] $2.5 \le \sup\left\{\frac{M_{ECT}(\mathcal{T},\mathcal{P})}{M_{OPT}(\mathcal{T},\mathcal{P})}\right\} \le 3 - \frac{2}{m}$. ■

4. References

[1] K.P. Belkhale and P. Banerjee, Approximate Algorithms For The Partitionable Independent Task Scheduling Problem, *Proc. International Conf. on Parallel Processing*, August 1990, Vol.I, pp.72-75.

[2] J. Blazwicz, M. Drabowski, and J. Weglarz, Scheduling Multiprocessor Tasks to Minimize Schedule Length, *IEEE Transactions on Computers*, Vol.c-35, 1986, pp.389–393.

[3] J. Du and J. Y-T. Leung, Complexity of Scheduling Parallel Task Systems, *SIAM Journal on Discrete Mathematics*, Vol.2, 1989, pp.473–487.

[4] R. L. Graham, Bounds on Multiprocessing Timing Anomalies, *SIAM Journal on Applied Mathematics*, Vol.17, 1969, pp.416–429.

[5] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, *Annals of Discrete Mathematics*, Vol. 15, 1979, pp.287–326.

[6] Q. Wang and K. H. Cheng, List Scheduling of Parallel Tasks, to appear in *Information Processing Letters*.

[7] Q. Wang and K. H. Cheng, A Heuristic of Scheduling Parallel Tasks and Its Analysis, *Technical Report #UH-CS-90-28, Computer Science Dept., University of Houston*, July 1990, 20pp.

Scheduling Precedence Graphs to Minimize Completion Time in Partitionable Parallel Architectures

Hyeong-Ah Choi and Bhagirath Narahari
Dept. of Electrical Engineering & Computer Science
The George Washington University
Washington, DC 20052

The problem of non-premptively scheduling a set of partially ordered tasks, represented by a precedence graph, on a multiprocessor has been extensively studied in the past. Recently, there have been studies that address the problem of scheduling tasks on parallel processing systems with interprocessor communication times [3], and the problem is shown to be NP-complete. Tasks are represented by a precedence graph and each node of the precedence graph, which represents a sub-task that must be completed before its successor can commence execution, is allocated to a single processor. In this paper we consider the problem of scheduling the task graph on partitionable parallel architectures, where each graph node represents a sub-task that in turn is a parallel task and can be mapped to a set of processors. A partitionable parallel processing system consists of a pool of processors, controllers and common system resources, and can be configured into a number of simultaneous partitions each of which executes a sub-task in parallel. Thus, two issues are introduced in partitionable systems: firstly, each of the sub-tasks can be performed using one of a number of parallel algorithms running on different configurations and and secondly, there is an overhead time to move data, i.e., the reconfiguration time, from the configuration required by one algorithm to the configuration required by the algorithm of the successor sub-task. The execution time for each algorithm and the reconfiguration times between all pairs of algorithms are provided along with the precedence graph. Thus the first step in the scheduling process, would be to select the algorithms (that would minimize the completion time) which have to be scheduled and this is the problem addressed in this paper.

To represent our problem, the precedence graph must be logically extended to what we call a *communication graph.* Since each node representing a sub-task can be implemented using a number of alternative algorithms, it logically consists of a number of nodes each of which has a weight denoting the total execution time of the corresponding algorithm. Each edge is now logically composed of a number of edges between the alternative algorithms of the two sub-tasks. The weights associated with these logical edges give the data movement time required between the two alternative algorithms for the two sub-tasks.

The goal in this paper is to determine what alternative algorithms, *i.e.* , configurations, to select so as to minimize the task completion time. We note that once the algorithms are selected, to complete the scheduling process the selected algorithms, represented by a precedence graph, must be passed to schedulers that would allocate and schedule partitions of processors. We are investigating such schedulers. The problem of scheduling optimal configurations can now be stated as follows: "*Given a task precedence graph, along with alternative algorithms and their execution times for each sub-task and the data movement times between algorithms of dependent sub-tasks, find for each sub-task the algorithm to be executed such that the completion time for the task, which includes the execution and data movement times, is minimized.*"

As an input to the problem of scheduling optimal configurations, we are given a task precedence graph, alternative algorithms and their execution times for each sub-task, and the data movement times between algorithms of dependent sub-tasks. Let

G_0 be the task precedence graph and to represent other information we will define a communication graph G_1 as follows. We assume that there are n sub-tasks, i.e., $\mid V(G_0) \mid= n$, $V(G_0) = \{v_1, v_2, \ldots, v_n\}$ corresponding to subtasks $T_1, T_2, \ldots T_n$, and each sub-task T_i has k_i choices of algorithms that can be executed. The vertex set of G_1 is defined to be $\bigcup_{1\leq i\leq n} V_i$, where $V_i = \{v_i^1, v_i^2, \ldots, v_i^{k_i}\}$ corresponds to sub-task T_i such that v_i^p represents the p-th choice of algorithm for T_i. The edge set of G_1 is defined such that $E(V_i, V_i) = \emptyset$, and for any two sets V_i and V_j, $1 \leq i, j \leq n$, either $E(V_i, V_j) = \emptyset$ if $(v_i, v_j) \notin E(G_0)$ or $E(V_i, V_j) = \{(v_i^p, v_j^q) \mid 1 \leq p \leq k_i \text{ and } 1 \leq q \leq k_j\}$ if $(v_i, v_j) \in E(G_0)$, where $E(V_i, V_j)$ represents the set of edges whose tails are in V_i and whose heads are in V_j. With each vertex $v_i^p \in V(G_1)$ there is associated a positive number $w(v_i^p)$ representing the execution time required by the p-th choice for sub-task T_i. Further, to each edge $e = (v_i^p, v_j^q) \in E(G_1)$, there is assigned a positive number $w(e)$ representing the data movement time required to send the output of the p-th choice for T_i to the input of the q-th choice for T_j. Let $\Delta(G)$ denote the maximum degree (sum of indegree and outdegree) among all nodes in G.

For any subgraph $H \subseteq G_1$ that is isomorphic to G_0, let P_H be a directed path in H and define $W(V(P_H)) = \sum\{w(v) \mid v \in V(P_H)\}$, $W(E(P_H)) = \sum\{w(e) \mid e \in E(P_H)\}$, and $W(P_H) = W(V(P_H)) + W(E(P_H))$. Note that $W(V(P_H))$ denotes the sum of the weights of the vetices in the path and $W(E(P_H))$ denotes the sum of the weights of the edges in the path.

Let $W_{\max}(H)$ denote the weight of the path with the largest weight in subgraph H, *i.e.*, $W_{\max}(H) = \max\{W(P_H) \mid P_H \text{ is a directed path in } H\}$.

Our problem is now defined as: *Find a subgraph H of G_1 that is isomorphic to G_0 such that $W_{\max}(H)$ is as small as possible.*

In what follows we state our results whose details are discussed in [1]. It can be seen that Theorems 1 and 2 close the gap between the polynomial time and NP-complete cases.

Theorem 1.*Given a precedence graph G_0 and a communication graph G_1, it is NP-hard to find a subgraph H of G_1 such that H is isomorphic to G_0 and $W_{\max}(H)$ is as small as possible, even if $\Delta(G_0) = 3$.*

Theorem 2. *Given a precedence graph G_0 with $\Delta(G_0) \leq 2$ and a communication graph G_1, there exists an $O(\mid V(G_1) \mid^3)$ algorithm to find a subgraph H of G_1 such that H is isomorphic to G_0 and $W_{\max}(H)$ is as small as possible.*

Theorem 3: *Given a precedence graph G_0 represented by a tree (i.e., G_0 does not have any cycle) and a communication graph G_1, there exists an $O(\mid E(G_1) \mid)$ algorithm to find a subgraph H of G_1 such that H is isomorphic to G_0 and $W_{\max}(H)$ is as small as possible.*

Remarks: The same problem problem with the objective of minimizing the total system time, which maximizes throughput, is discussed in [2].

References

[1] H.-A. Choi, and B. Narahari, "Scheduling Precedence Graphs on Partitionable Parallel Architectures" Technical Report, Dept. of Electrical Engineering and Computer Science, The George Washington University, September 1990.

[2] H.-A. Choi, and B. Narahari, "Scheduling Precedence Graphs to Minimize Total System Time on Partitionable Parallel Architectures" to appear in *2nd IEEE Symposium on Parallel and Distributed Processing,* Dallas, Texas, December 1990.

[3] Jing-Jang Hwang, Yuan-Chieh Chow, Frank D. Angers, and Chung-Yee Lee, "Scheduling Precedence Graphs in Systems with Interprocessor Communication Times" *SIAM Journal of Computing* , Vol.18, No.2, pp.244–257, April 1989.

AN *NC* APPROXIMATION ALGORITHM FOR THE MINIMUM DEGREE SPANNING TREE PROBLEM

Martin FÜRER
Computer Science Department, Pennsylvania State University
333 Whitmore Building, University Park, PA 16802
furer@cs.psu.edu

Balaji RAGHAVACHARI
Computer Science Department, Pennsylvania State University
333 Whitmore Building, University Park, PA 16802
rbk@cs.psu.edu

Abstract. Computing a spanning tree of a graph that satisfies given constraints has been studied before and the problems have been classified into different complexity classes. We look at the problem of constructing a spanning tree for a graph $G = (V, E)$, whose maximum degree is the smallest among all possible spanning trees. This problem is easily shown to be *NP*-hard. We describe an *NC* algorithm that computes a spanning tree of a graph, whose degree is within a factor of $O(\log n)$ of the degree of an optimal tree. The algorithm is then extended to directed graphs to compute a rooted spanning tree of a directed graph, whose indegree is within a factor of $O(\log n)$ of the indegree of an optimal rooted tree.

1. Introduction

The problem of computing a spanning tree of a graph that satisfies given constraints has been studied before. Many constraints have been considered in [1, 2, 3] and the problems have been classified into different complexity classes. A comprehensive list of *NP*-complete constrained spanning tree problems can be found in [4, 7].

We consider the problem of constructing a spanning tree for a graph $G = (V, E)$, whose maximum degree is the smallest among all possible spanning trees. Consider a spanning tree T of G. Let Δ be the degree of T. We want to find some tree T^* with degree Δ^* such that

$$\Delta^* = \min_T(\Delta)$$

where T ranges over all spanning trees of G. This can immediately be shown to be *NP*-Hard by reducing the Hamiltonian Path Problem to the minimum degree spanning tree problem. The language version of this problem is, given a graph $G = (V, E)$ and an integer k, to decide if there exists a spanning tree of G whose maximum degree is bounded by k. This problem is *NP*-complete [4].

This problem arises in the following context. The question of whether Depth First Search is in deterministic *NC* has been open for quite some time. One approach that seems to be worth investigating is to start with some spanning tree of the graph and try to modify it into a depth first search tree. Note that modifications to the tree can be

---

This work was supported in part by the National Science Foundation under grant CCR-8805978.

defined locally in the neighborhood of each vertex, thus making it possible to parallelize the algorithm. One of the obvious stumbling blocks to this approach is the case when the initial spanning tree has vertices of unnecessarily high degree. This algorithm tries to choose a *nice* initial spanning tree. Its application to Depth First Search is yet to be explored.

As the minimum degree spanning tree problem turns out to be *NP*-hard, there is not much hope of obtaining an efficient deterministic algorithm to solve it in the sequential case and even less chance of getting an *NC* algorithm to solve the problem. The next logical step is to try to find an approximate solution to the above problem. The idea here is to try to construct some spanning tree T' whose degree is Δ' in such a way that some bound can be placed on $\frac{\Delta'}{\Delta^*}$. In this paper, we show that we can find some spanning tree T' such that

$$\frac{\Delta'}{\Delta^*} = O(\log n)$$

The most interesting part is that the approximation algorithm is easily parallelizable. In other words, we describe an approximation algorithm in *NC* for an *NP*-complete problem. The algorithm is then generalized to find rooted spanning trees for directed graphs.

It is well known that the problem of finding a spanning tree (in particular, a minimum weight spanning tree) is a matroid problem and that the greedy algorithm is easily parallelizable [9]. The algorithm keeps a spanning forest of the graph and works in phases. In each phase, each connected component of the spanning forest finds another component to which it is connected in the graph. This is followed by merging these components into one. This algorithm does not try to restrict the degree of the spanning tree in any way. It is possible that in the first phase, all vertices may choose vertex 1 as their partner and we get a spanning tree in one phase. In this case the degree of vertex 1 would be $(n-1)$. It is not obvious how this algorithm can be adapted to get a spanning tree of as small a degree as possible.

2. Ideas behind the approximation algorithm

Let the given graph be $G = (V, E)$ with n vertices. It is assumed that the graph G is connected (in the case of directed graphs, it is assumed that there is some vertex r which is reachable from all other vertices). Let T^* be an optimal spanning tree whose maximum degree is Δ^*. During phase i of the algorithm, we have a spanning forest F_i containing f_i connected components. Let $G_{F_i} = (V_{F_i}, E_{F_i})$ be a graph constructed as follows. V_{F_i} has f_i vertices, and each vertex represents a connected component of F_i. The edge set E_{F_i} is obtained by having an edge between two vertices whenever the corresponding components have an edge connecting them in G. The algorithm starts with each vertex of the graph G as a separate component, thus forming F_1.

We introduce the following key ideas that modify the greedy algorithm to find a spanning tree whose degree is within a factor of $O(\log n)$ of the optimal spanning tree. During the choosing phase, the greedy algorithm places no restriction on the selection of partners. This makes it possible for all components to choose one particular component as their respective partners. Due to this, the degrees of the vertices in *that* component could become much larger than is necessary. To avoid this, we replace the choosing phase by computing a maximal matching of G_{F_i}. The components across matched edges are merged in the next step. But this alone does not solve the problem because the size of the maximal matching may be very small, thus increasing the number of phases to $\Omega(n)$, in the worst case. To manage the case when the size of the matching is too small to be acceptable, we generalize the idea of a matching to a $(k,1)$-*matching* (defined later). Note that all the unmatched components are connected only to the matched components.

Let the maximal matching split the vertices of G_{F_i} into the matched components V_F^M and the unmatched components V_F^U. Construct a bipartite graph $B_F = (X, Y, E_B)$ defined as follows. The vertex set X contains those vertices of the original graph G that form V_F^M. The vertex set Y equals the set V_F^U, the set of unmatched components. The edge set E_B is obtained by having an edge from $x \in X$ to $y \in Y$ whenever the vertex x is connected by an edge to one of the vertices in the component y in the original graph G.

We now define a $(k,1)$-matching of the graph B_F as a subgraph of B_F in which the degree of the vertices in X is at most k and the degree of vertices in Y is at most 1. In other words, the vertices in X are allowed to match up to k vertices from Y (polygamy of size k for all $x \in X$). The *size* of a $(k,1)$-matching is defined to be the number of matched vertices of Y. A *maximal* $(k,1)$-matching is defined as a $(k,1)$-matching which is not properly contained in another $(k,1)$-matching.

With these definitions, a matching in a bipartite graph is an $(1,1)$-matching and a maximal matching is a maximal $(1,1)$-matching. Note that the maximal $(k,1)$-matching problem is easily reduced to the maximal matching problem as follows. Construct a new graph by making k copies of the vertices in X and for each of these copies, connect all adjacent vertices from the original bipartite graph. Any maximal matching in the new graph corresponds to a maximal $(k,1)$-matching in the original graph.

3. The approximation algorithm

```
Procedure Approximate-Minimum-Degree-Spanning-Tree($G$)
  (* This procedure computes an approximation for a minimum-degree spanning tree of a graph $G$ *)
  1. $F \leftarrow \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$;   $f \leftarrow n$;   $k \leftarrow 1$
  2. If $f = 1$ then stop.
  3. Compute the graph $G_F$ and find a maximal matching $M$ in it. Let $|M| = m$.
  4. Using the matched components $V_F^M$, and the unmatched components $V_F^U$, construct the
       bipartite graph $B_F$ as described earlier.
  5. Repeat forever
       5.1 Find $M_k$, a maximal $(k,1)$-matching in $B_F$. Let $|M_k| = m_k$.
       5.2 If $m_k \geq f/2 - m$ then combine the components across matched edges in $M$ and $M_k$,
             update $F$ and $f$ and go to step 2.
       5.3 $k \leftarrow k * 2$
     End Repeat
End Procedure
```

Algorithm 1. Computing an Approximation for the Minimum Degree Spanning Tree

Each execution of steps 2-5 is considered as a phase. It is obvious that the algorithm correctly constructs a spanning tree. To analyze the running time, observe that in each phase, the number of connected components f, goes down by at least a half. Therefore there are at most $O(\log n)$ phases. Using this, we will show subsequently that maximal matchings are computed at most $O(\log n)$ times. Hence the running time of the algorithm is $O(M(n) \cdot \log n)$, where $M(n)$ is the running time of the maximal matching algorithm. We now show the bound on the degree of the resulting spanning tree.

Lemma 3.1: In step 5.1 of Algorithm 1, if $k \geq \Delta^*$, then the size of every maximum $(k, 1)$-matching in B_F is $|Y| = f - 2m$. Hence the size of any maximal $(k, 1)$-matching is at least $f/2 - m$.

Proof: Consider a spanning tree T^* of degree Δ^* in G. As T^* is a spanning tree, it is possible to find a set of edges $W \in T^*$ which is a spanning tree in G_F. By looking at the edges in B_F that correspond to W, we can show that there exists a $(k, 1)$-matching that matches all elements of Y. As T^* is a spanning tree, it must be the case that each component $y \in Y$ is connected to at least one of the vertices $x \in X$ by an edge in T^* (note that there are no edges between the components of Y). For each y, pick one of these vertices arbitrarily and match y to x. This matches all the components of Y and the degree of the vertices in X is bounded by Δ^*. Therefore there exists a $(\Delta^*, 1)$-matching whose size is $f - 2m$. For $k \geq \Delta^*$, this is also a $(k, 1)$-matching.

The second statement follows from the fact that a maximal matching of a graph is at least half the size of a maximum matching. Having reduced the maximal $(k, 1)$-matching problem to the maximal matching problem, this follows immediately. ∎

From the above lemma, we can see that in step 5 of the algorithm, the iteration stops as soon as k gets greater or equal to Δ^*. From this the main theorem follows directly.

Theorem 3.2: Algorithm 1 finds a spanning tree of maximum degree Δ such that

$$\frac{\Delta}{\Delta^*} = O(\log n)$$

Proof: By Lemma 3.1, the algorithm exits the loop in step 5 when $k < 2\Delta^*$. In each phase, the degree of any vertex increases by at most $(k + 1)$. Therefore the degree of any vertex goes up by at most $2\Delta^*$ in each phase. The rest of the result follows from the observation that there are at most $O(\log n)$ phases to the algorithm. ∎

Theorem 3.3: Algorithm 1 runs in $O(M(n) \cdot \log n)$, where $M(n)$ is the time required to compute a maximal matching in parallel.

Proof: We first show that maximal matchings are computed only $O(\log n)$ times by the algorithm. Assuming that each computation of a maximal matching costs 1 unit, we distribute the cost as follows. A maximal matching is computed in step 3 and a cost of 1 is charged to the corresponding phase. Maximal matchings may be computed more than once in step 5. In each phase, the cost of the first of these is charged to the phase. The remaining charges are counted separately. Each phase gets the charge for the computation of two maximal matchings. As there are $O(\log n)$ phases, this generates a cost of $O(\log n)$. By Lemma 3.1, k stops growing as soon as it gets greater or equal to Δ^*. Therefore the number of maximal matchings that have not been charged above is $O(\log \Delta^*)$. Combining the two, maximal matchings are computed $O(\log n)$ times by the algorithm. As the computation of maximal matchings is the most time consuming part of the algorithm, it runs in $O(M(n) \cdot \log n)$ time. ∎

The only non-trivial part of the algorithm is the computation of maximal matchings. So it is easily parallelized into an *NC* algorithm by using the efficient algorithm of Israeli and Shiloach for computing maximal matchings [6]. This algorithm computes a maximal matching on a *CRCW* PRAM using $|V| + |E|$ processors in $O(\log^3 |E|)$ time, which is $O(\log^3 n)$. Note that we can also use Luby's maximal independent set algorithm for computing maximal matchings [8]. This algorithm runs on a *EREW* PRAM in $O(\log^2 n)$ but uses a larger number of processors.

Corollary 3.4: With the Israeli and Shiloach's algorithm for computing maximal matchings, the algorithm finds an approximation within a factor of $O(\log n)$ for the minimum degree spanning tree using $|E| + |V|$ processors in $O(\log^4 n)$ time on a *CRCW* PRAM. ∎

Corollary 3.5: With Luby's algorithm for computing maximal matchings, the algorithm finds an approximation within a factor of $O(\log n)$ for the minimum degree spanning tree in NC^3. ∎

4. Extending the algorithm to directed graphs

In the case of directed graphs, the above algorithm does not work due to the following reason. Consider the arc (q, p) that merges two subtrees P and Q. Let $p \in P$ and $q \in Q$. In the case of undirected graphs, any two subtrees can be merged using any arbitrary edge connecting the two. But in the case of directed graphs, the vertex q must be the root of the subtree Q. Otherwise the procedure would not yield a rooted spanning tree of G.

To handle this problem, the following modifications are made to the algorithm. In the construction of the graph G_{F_i}, two components were connected whenever there was an edge in G between the corresponding connected components. Let Q be one of the subtrees in F_i with root q. Let S_Q be the set of vertices of Q that are in the strongly connected component of q, in the subgraph of G induced by the vertices in Q. Note that the rooted subtree Q can easily be modified to yield another rooted subtree on the same vertices with any of the vertices of S_Q as its root. This can be done in such a way that the indegree of any vertex in Q increases by at most one (left as an exercise to the reader).

Now the construction of the graph G_{F_i} is modified as follows. The vertices of G_{F_i} correspond to the rooted subtrees of F_i. An edge is drawn between vertices P and Q

in G_{F_i} whenever there is an arc (y, x) such that either

$$x \in P \quad \text{and} \quad y \in S_Q, \quad \text{or}$$
$$x \in Q \quad \text{and} \quad y \in S_P.$$

It is obvious that an edge between P and Q implies that the corresponding rooted trees can easily be merged into one.

The construction of the bipartite graph $B_F(X, Y, E_B)$ is also modified as follows. The vertex set X is formed as before and contains the vertices of G that form V_F^M. The vertex set Y is also formed as before, with each vertex representing one rooted subtree from V_F^U. But now, an edge is drawn between $x \in X$ and $y \in Y$ only when there is an arc (w, x) in G such that $w \in S_y$.

```
Procedure Approximate-Minimum-Degree-Rooted-Spanning-Tree($G$)
  (* This procedure computes an approximation for a minimum-degree rooted spanning tree of a
     directed graph $G$ *)
  1. $F \leftarrow \{\{v_1\}, \{v_2\}, ..., \{v_n\}\}$;  $f \leftarrow n$;  $k \leftarrow 1$
  2. If $f = 1$ then stop.
  3. Using the strongly connected components of the roots of the subtrees in $F$, compute
       the graph $G_F$ and find a maximal matching $M$ in it. Let $|M| = m$.
  4. Using the matched components $V_F^M$, and the unmatched components $V_F^U$, construct the
       bipartite graph $B_F$, with the modifications for directed graphs as described earlier.
  5. Repeat forever
       5.1 Find $M_k$, a maximal $(k, 1)$-matching in $B_F$. Let $|M_k| = m_k$.
       5.2 If $m_k \geq (f - 1)/2 - m$ then
           5.2.1 Move the root of matched components in $Y$, if necessary.
           5.2.2 Combine the components across matched edges in $M$ and $M_k$, update $F$ and $f$
                 and go to step 2.
       5.3 $k \leftarrow k * 2$
     End Repeat
End Procedure
```

Algorithm 2. Approximation Algorithm for Directed Graphs

Algorithm 2 is given above and includes all the changes that were discussed earlier. Also step 5.2 has been modified so that the program exits the repeat loop in step 5 if $m_k \geq (f-1)/2 - m$. As before the algorithm clearly computes a rooted spanning tree, if one exists. The running time of the algorithm also stays the same. But the number of processors increases, because in each phase the strongly connected components have to be computed for each rooted subtree.

The following lemma shows that the growth of the degree of any vertex in each phase is bounded by $O(\Delta^*)$. Note that it is analogous to Lemma 3.1.

Lemma 4.1: In step 5.1 of Algorithm 2, if $k \geq \Delta^*$, then the size of every maximum $(k,1)$-matching in B_F is at least $|Y| - 1 = f - 2m - 1$. Hence the size of any maximal $(k,1)$-matching is at least $(f-1)/2 - m$.

Proof: As before, we show that whenever $k \geq \Delta^*$, there exists a $(k,1)$-matching that has the desired size by looking at some optimal tree T^* with indegree Δ^*. Let r^* be the root of T^*. Observe that any arbitrary set of vertices that does not contain r^* must have an outgoing arc in T^* (because T^* is a rooted spanning tree). For each $y \in Y$ that does not contain r^*, consider the set of vertices S_y, as discussed before. S_y contains the set of vertices that are in the strongly connected component of the root of y in the subgraph of G induced by the vertices in y. By the above observation, there is at least one arc (w,x) in T^*, such that $w \in S_y$. The graph was partitioned into X and Y based on a maximal matching. Therefore there are no edges between vertices in Y. Hence it is necessary that $x \in X$. For each $y \in Y$ that does not contain r^*, arbitrarily choose one of these outgoing arcs that belongs to T^*, and use it in the $(k,1)$-matching.

If the rooted subtree containing r^* is in Y, then it may not be possible to match this component. This is the reason why we may get $f - 2m - 1$ matching edges instead of $f - 2m$ as in the undirected case. We have shown that there is a $(\Delta^*,1)$-matching in B_F of the required size. For $k \geq \Delta^*$, this is also a $(k,1)$-matching and the lemma follows. ▮

Theorem 4.2: Algorithm 2 finds a rooted spanning tree of maximum indegree Δ such that

$$\frac{\Delta}{\Delta^*} = O(\log n)$$

Proof: The proof of this theorem is similar to that of Theorem 3.2. By Lemma 4.1, each phase of the algorithm increases the indegree of any vertex by at most $2\Delta^*$. Also in each phase the number of connected components f, reduces by at least $(f-1)/2$ which is at least $f/3$, for $f \geq 3$. Therefore there are at most $O(\log n)$ phases. Combining these facts, we get the desired result. ▮

The algorithm requires the computation of strongly connected components in each phase. This increases the number of processors required from $O(|E|)$ to $O(S(n))$, where $S(n)$ is the number of processors to compute the strongly connected components of a given graph in $O(\log^2 n)$ time.

Corollary 4.3: With the Israeli and Shiloach's algorithm for computing maximal matchings, the algorithm finds an $O(\log n)$ approximation for the minimum degree rooted spanning tree of a directed graph using $O(S(n))$ processors in $O(\log^4 n)$ time on a *CRCW* PRAM. ▮

Corollary 4.4: With Luby's algorithm for computing maximal matchings, the algorithm finds an approximation within a factor of $O(\log n)$ for the minimum degree rooted spanning tree in NC^3. ▮

5. Conclusion

An *NC* approximation algorithm has been demonstrated for the minimum degree spanning tree problem. It would be interesting to see whether a constant factor approximation is possible for this problem. Also the running time of our algorithm depends on the time taken by the maximal matching algorithm. Maximal matchings are very fundamental to many parallel graph algorithms, but there is a significant gap in the running time of deterministic and randomized algorithms. The best deterministic *NC* algorithm for computing a maximal matching was designed by Israeli and Shiloach and runs in $O(\log^3 n)$ time using $O(|E|)$ processors [6]. But Israeli and Itai have given an *RNC* algorithm which runs in only $O(\log n)$ time using $O(|E|)$ processors [5]. Reducing the deterministic parallel complexity of maximal matching can reduce the running time of our algorithm. Not much is known about most of the other related *NP*-complete constrained spanning tree problems. Some open problems would be to find approximation algorithms for these and also find more applications for them.

References

[1] P.M. Camerini and G. Galbiati, *The Bounded Path Tree Problem, SIAM J. Algeb. Disc. Meth.* **3**, (1982) 474-484.

[2] P.M. Camerini, G. Galbiati and F. Maffioli, *Complexity of Spanning Tree Problems: Part I, European J. Oper. Res.* **5** (1980) 346-352.

[3] P.M. Camerini, G. Galbiati and F. Maffioli, *On the Complexity of Finding Multi-constrained Spanning Trees, Disc. Appl. Math.* **5** (1983) 39-50.

[4] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman, 1979.

[5] A. Israeli and A. Itai, *A Fast and Simple Randomized Parallel Algorithm for Maximal Matching, Inf. Proc. Lett.* **22** (1986) 77-80.

[6] A. Israeli and Y. Shiloach, *An Improved Parallel Algorithm for Maximal Matching, Inf. Proc. Lett.* **22** (1986) 57-60.

[7] D.S. Johnson, *The NP-completeness Column: An Ongoing Guide, J. Algorithms* **6** (1985) 145-159.

[8] M. Luby, *A Simple Parallel Algorithm for the Maximal Independent Set Problem, SIAM J. Comput.* **15** (1986) 1036-1053.

[9] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall (1982).

Optimal Parallel Algorithms for Problems Modelled by a Family of Intervals

Stephan Olariu[†][‡], James L. Schwing[†], Jingyuan Zhang[†]

Department of Computer Science
Old Dominion University
Norfolk, VA 23529-0162
U.S.A.

Abstract. A family of intervals on the real line provides a natural model for a vast number of scheduling and VLSI problems. Recently, a number of parallel algorithms to solve a variety of practical problems on such a family of intervals have been proposed in the literature. We develop a new computational technique and show how it can be used for the purpose of devising cost-optimal parallel algorithms for a number of interval-related problems including finding a largest subset of pairwise non-overlapping intervals, a minimum dominating subset of intervals, along with algorithms to compute the shortest path between a pair of intervals and, based on the shortest path, a parallel algorithm to find the center of the family of intervals. More precisely, with an arbitrary family of n intervals as input, all our algorithms run in O(log n) time using O(n) processors in the EREW-PRAM model of computation.

1. Introduction

We associate a graph $G = (V,E)$ with the family $I = \{I_i = [a_i, b_i] \mid a_i \leq b_i,\ 1 \leq i \leq n\}$ of intervals on the real line, the vertices of G are precisely the intervals in I, with two vertices adjacent if, and only if, the corresponding intervals overlap. In this context, G is commonly referred to as the *intersection graph* of I or, simply, the *interval graph* corresponding to I. Conversely, I is termed the *interval model* of the graph G.

We assume the well known Parallel Random Access Machine (PRAM) model. In a Concurrent Read Exclusive Write PRAM (CREW-PRAM) model, several processors may simultaneously read the same memory location, but exclusive access is used for writing. In the Exclusive Read Exclusive Write PRAM (EREW-PRAM) model, a memory location cannot be simultaneously accessed by more than one processor. The interested reader is referred to [12] for a competent discussion on the PRAM family.

A subset C of vertices of a graph is a *clique* if every two vertices in C are adjacent. Similarly, a subset S of vertices of a graph is *independent* if every two vertices in S are non-adjacent. A set D of vertices of G is said to be *dominating* if every vertex outside D is adjacent to a vertex in D.

Given an arbitrary graph G, a sequence of vertices $[v_0, v_1, v_2, \ldots, v_t]$ is a *path* joining v_0 to v_t if v_{i-1} and v_i are adjacent for all $i = 1, 2, \ldots, t$. The parameter t is termed the *length* of the path. We note that, in case G is connected, there is at least one path joining every pair of vertices in G. A *shortest path* is the one which has minimal length. In this context, the *distance* $d_G(u,v)$ (or $d(u,v)$ for short) between vertices u and v is the length of the shortest path from u to v. Using this, the *diameter*, *radius*, and *center* of G are defined as:

$$\mathrm{diam}(G) = \max_{u,v \in V} d_G(u,v),\quad r(G) = \min_{u \in V} \max_{v \in V} d_G(u,v),\ \text{and}\ C(G) = \{u \in V \mid \max_{v \in V} d_G(u,v) = r(G)\}.$$

Recently, there has been a considerable interest in devising efficient parallel algorithms for interval graphs. Dekel and Sahni [5,6] have proposed an efficient algorithm to solve the *channel assignment* problem. Bertossi and Bonucelli [1] have proposed a number of parallel algorithms to solve various optimization problems in the context of interval graphs: they devised O(log n) algorithms using $O(n^2 / \log n)$ processors in the CREW-PRAM model to compute a maximum weight clique, a minimum clique cover, a maximum independent set, a minimum size dominating set, as well as computing a hamiltonian circuit and the bandwidth of a proper interval graph (see [1,8] for details and relevant definitions). They have also devised $O(\log^2 n)$ algorithms using $O(n^3 / \log n)$ processors in the CREW-PRAM model to compute a maximum weight independent set and a minimum weight dominating set in an interval graph, while leaving open the problem of obtaining more efficient algorithms.

[†] This work was supported by NASA under grant NCC1-99

[‡] Additional support by the National Science Foundation under grant CCR-8909996 is gratefully acknowledged .

The purpose of this work is to present a number of *cost-optimal* parallel algorithms which, given an interval graph G specified by its interval model, solve the problems of computing: a maximum size independent set, a minimum size clique cover, a minimum size dominating set, a shortest path between any pair of vertices, and the center of G.

It is worth mentioning that all our algorithms rely on techniques which are of an independent interest and which we shall present as new results in their own right. The paper is organized as follows: Section 2 presents a number of lower bound results of import in our optimality arguments; Section 3 gives the details of the techniques used in our parallel algorithms; finally, Section 4 presents our cost-optimal parallel algorithms for an arbitrary family of intervals on the real line.

2. Lower Bounds

Our optimality arguments rely, in part, on the fact that the following problems have a lower bound of $\Omega(n \log n)$ in the algebraic tree model of computation: ELEMENT UNIQUENESS, NON-OVERLAPPING INTERVALS, UNIFORM GAP and MAXIMUM GAP. We refer the reader to [11] for details and derivations of the above problems.

To simplify the notation, let $\mu(X)$ stand for the maximum gap of a set $X = \{x_1, x_2, ..., x_n\}$. In addition, define the following problems and prove that their lower bound is also $\Omega(n \log n)$.

- MAXIMUM BRIDGE. Given a set X of n real numbers $x_1, x_2, ..., x_n$ and a real $\delta > 0$, determine whether $\delta \geq \mu(X)$.

Lemma 1. MAXIMUM BRIDGE has a lower bound of $\Omega(n\log n)$.

Proof. The result follows by reducing UNIFORM GAP to MAXIMUM BRIDGE. □

- INTERVAL CONNECTIVITY. Given an n-vertex interval graph G specified by its interval model, decide if G is connected.

Lemma 2. INTERVAL CONNECTIVITY has a lower bound of $\Omega(n \log n)$.

Proof. The result follows by reducing MAXIMUM BRIDGE to INTERVAL CONNECTIVITY. □

In our arguments, it will be convenient to rely on the following parameters of the family I:

$$\text{first}(I) = I_j \text{ with } b_j = \min\{b_i \mid 1 \leq i \leq n\}; \text{ last}(I) = I_k \text{ with } a_k = \max\{a_i \mid 1 \leq i \leq n\}.$$

That is, first(I) is the interval of the family whose right-endpoint is farthest to the left and last(I) is the interval whose left-endpoint is farthest to the right. It is worth noting that by (1.3), described in the next section, both first(I) and last(I) can be computed in $O(\log n)$ time using $\lceil n / \log n \rceil$ processors in the EREW-PRAM model.

- SHORTEST PATH. Given an n-vertex interval graph G specified by its interval model, and arbitrary vertices u, v in G, find a shortest path joining u and v, if such a path exists.

Lemma 3. SHORTEST PATH has a lower bound of $\Omega(n \log n)$.

Proof. The result follows by reducing INTERVAL CONNECTIVITY to SHORTEST PATH. □

- CENTER. Given an n-vertex interval graph G, compute the center of G.

Lemma 4. CENTER has a lower bound of $\Omega(n \log n)$.

Proof. The result follows by reducing INTERVAL CONNECTIVITY to CENTER.□

Finally, we conclude this section by considering two more functions of interval families and their corresponding computation problems. As in [1], define for every interval I_i:

$$\text{right}(I_i) = \{I_j, \text{ if } b_j = \max\{b_k \mid a_k \leq b_i\} \ \&\ b_j > b_i \ \| \ \text{nil,otherwise}\} \text{ and}$$

$$\text{next}(I_i) = \{I_j, \text{ if } b_j = \min\{b_k \mid b_i < a_k\} \ \| \ \text{nil,otherwise}\} .$$

That is, $\text{right}(I_i)$ is the interval ending farthest to the right among all intervals that overlap with with I_i, if such an interval exists. Similarly, $\text{next}(I_i)$ is the interval that ends farthest to the left among all the intervals beginning after the end of I_i.

- RIGHT. Given a family $I = \{I_i = [a_i,b_i] \mid a_i \leq b_i, 1 \leq i \leq n\}$ of intervals on the real line, compute right(I_i) for all i ($1 \leq i \leq n$).

Lemma 5. RIGHT has a lower bound of $\Omega(n \log n)$.

Proof. The result follows by reducing INTERVAL CONNECTIVITY to RIGHT. □

- NEXT. Given a family $I = \{I_i = [a_i,b_i] \mid a_i \leq b_i, 1 \leq i \leq n\}$ of intervals on the real line, compute next(I_i) for all i ($1 \leq i \leq n$).

Lemma 6. NEXT has a lower bound of $\Omega(n \log n)$.

Proof. The result follows by reducing ELEMENT UNIQUENESS to NEXT. □

3. The Techniques

The problem of *list ranking* is to determine in parallel the rank of every element in a given linked list (stored as an unordered array), that is, the number of elements following it in the list. Once every element knows its rank, an ordered array of the elements in the list can be constructed by simply using each rank as an index in the array. The list ranking problem generalizes easily to the *weighted rank* problem: here every element of the list has an associated weight and the problem is to determine the weighted rank of each element, that is the sum of weights of all subsequent elements in the list. List ranking has turned out to be one of the fundamental techniques in parallel processing, playing a crucial role in a vast array of important parallel algorithms (see [10] for details). In particular, Cole and Vishkin [4] have showed that list ranking and weighted list ranking can be done optimally in $O(\log n)$ time using $O(n / \log n)$ processors in the EREW-PRAM model.

Using list ranking the following problems can be solved optimally in $O(\log n)$ time using $O(n / \log n)$ processors in the EREW-PRAM:

(1.1) mark all the nodes in a linked list of (at most) n elements;

(1.2) given a linked list with certain elements marked, record the marked elements in an array in the order in which they occur in the list.

(1.3) given n real numbers $x_1, x_2, ..., x_n$, and an associative binary operation $\circ$ on $x_1, x_2, ...,$ x_n, compute all the *prefixes* of the form $x_1, x_1 \circ x_2, ..., x_1 \circ x_2 \circ ... \circ x_n$.

For our first technique consider the following problem.

- MULTIPLE SOURCE. Given an ordered set $x_1, x_2, ..., x_n$ with a subset $x_{i_1}, x_{i_2}, ..., x_{i_m}$ of the x_i's marked. The MULTIPLE SOURCE problem asks to transmit to each x_j ($1 \leq j \leq n$) the identity of the first marked element to the left of x_j (possibly x_j itself).

Lemma 7. The MULTIPLE SOURCE problem can be solved optimally in $O(\log n)$ time using $O(n / \log n)$ processors in the EREW-PRAM model.

Proof. For this purpose, assign to each x_i the label d_i such that $d_i = \{x_{i,}$ if x_i is marked $\mid\mid$ nil, otherwise$\}$. Next, define the following operation on the set $d_1, d_2, ..., d_n$, by putting for $i < j$: $d_i \circ d_j = \{d_j$, if $d_j \neq$nil $\mid\mid$ nil, otherwise$\}$. Checking that the operation $\circ$ is associative is straightforward and left to the reader. Finally, perform prefix $\circ$ on the d_i's and let $e_1, e_2, ..., e_n$ be the result. We note that for all $1 \leq j \leq n$ e_j contains x_k if, and only if, x_k is the first marked element "to" the "left" of x_j . To argue for the complexity, note that by (1.1)-(1.3) all the stages of this simple algorithm take $O(\log n)$ time using $O(n / \log n)$ processors. Since no read or write conflits occur, the computation can be performed in the EREW-PRAM model. □

Let T be a rooted, ordered tree with n nodes. For convenience, we assume that the tree is *standard form*, that is T is stored in an unordered array, with every node having a parent pointer along with a doubly linked list of children. The well-known *Euler-tour technique* developed in [12] allows one to compute various functions on trees including preorder number, postorder number, level of every node, and many others, by reducing all these problems to list ranking. To make our presentation self-contained, we shall present now the details of a variant of this technique. To begin, we replace every node u in T having d(u) children by an array U[1 .. d(u)+1] (of copies of u). Next, letting $W_1[1 .. d_1+1]$, $W_2[1 .. d_2+1]$, ..., $W_{d(u)}[1 .. d_{d(u)}+1]$ stand for the corresponding vectors of the children of u in T, for all i = 1, 2, ..., d(u), set: link(U[i]) $\leftarrow$ $W_i[1]$, link($W_i[d_i+1]$) $\leftarrow$ U[i+1], and finally, set link(U[d(u)+1]) $\leftarrow$ p(u), where, p(u) is the corresponding entry in the array of the parent of u in T.

Assuming that the root of T is R and that R has r children, what results is a linked list starting at R[1] and ending at R[r+1] with every edge of T traversed exactly once in each direction. Therefore, the total length of the resulting linked list is 2n−2. It is worth noting that for a tree T with n nodes, the linked list can be obtained in O(log n) time using O(n / log n) processors.

As an application of the Euler-tour technique we shall solve the following problem which will prove to be of a particular import in our parallel algorithms.

- PATH IDENTIFICATION. Given a rooted tree T and nodes u and v in T with v an ancestor of u, identify in parallel the unique path from u to v.

Lemma 8. Given a rooted n-node tree T in standard form and nodes u and v of T such that v is an ancestor of u, we can solve PATH IDENTIFICATION in O(log n) time using O(n / log n) processors in the EREW-PRAM model.

Proof. Assume that the tree T is represented in the standard form. To begin, we compute an Euler-tour of T; next, we compute the *level* of every node in T: assign to the first entry in every array a weight +1 and to all the others -1. It is easy to confirm that after performing a prefix sum, every node will receive a label that represents its level in T. Further, assign a weight of +1 to U[d(u)+1] and 0 to all the others. Now performing prefix sum again, we can isolate the sublist L starting at u and ending at the first entry in the array V (corresponding to the node v of T) which has received a value of 1 in this prefix sum. Once L is known, we perform a prefix minimum on the level of every node in L starting at u, marking the first occurence of every distinct value in the result. Obviously, what results is the unique path from u to the root of T. To argue for the complexity of this simple algorithm, we note that every step takes O(log n) time and at most $\lceil n / \log n \rceil$ processors using (1.1), (1.2), and (1.3), combined. Therefore, our algorithm is cost-optimal. □

Lemma 9. A rooted tree with n nodes specified by parent pointers can be put in standard form in O(log n) time using O(n) processors in the EREW-PRAM model. □

In [1], Bertossi and Bonuccelli solve the RIGHT and NEXT problem in *suboptimal* time: specifically, their algorithms run in O(log n) time using $O(n^2 / \log n)$ processors in the CREW-PRAM model. One of the contributions of this work is an optimal technique to solve both RIGHT and NEXT: to anticipate, we propose parallel algorithms to solve both problems in O(log n) time using O(n) processors in the EREW-PRAM model of computation. In the light of Lemma 5 and Lemma 6, these algorithms are optimal. In addition, it turns out that with our technique we can obtain a number of optimal algorithms for interval graphs.

```
Procedure Parallel_Find_Right({$I_i$});
{Input: a family of intervals {$I_i \mid 1 \le i \le n$}, where $I_i = [a_i, b_i]$.
 Output: right($I_i$) ($1 \le i \le n$)}
 begin
     let $c_1, c_2, ..., c_{2n}$ be the left and right-endpoints sorted in ascending order†;
     create $d_1, d_2, ..., d_{2n}$ with $d_i = \{b_j$,if $c_i$ is $a_j \mid\mid -\infty$,otherwise} ;
     perform prefix maximum on the sequence $d_1, d_2, ..., d_{2n}$, and
     let $e_1, e_2, ..., e_{2n}$ be the result (i.e. $e_i = \max\{d_1, d_2, ..., d_i\}$);
     let $I_{j_1}, I_{j_2}, ..., I_{j_{2n}}$ be the sequence such that $b_{j_i} = e_i$ ($1 \le i \le 2n$);
     for every $I_i$ ($1 \le i \le n$) do in parallel begin
         let k be the index of $b_i$ in the sequence $c_1, c_2, ..., c_{2n}$;
         if $e_k > b_i$ then
             right($I_i$) ← $I_{j_k}$
         else
             right($I_i$) ← nil
     end {for}
 end; {Parallel_Find_Right}
```

The following result asserts that Parallel_Find_Right optimally solves the RIGHT problem for a family of intervals on the real line.

---

† if $c_i = c_{i+1} = ...$, we assume that left-endpoints precede right-endpoints

Theorem 1. Procedure Parallel_Find_Right correctly solves the RIGHT problem in O(log n) time using O(n) processors in the EREW-PRAM model.

Proof. The correctness follows from the observation that after the prefix maxima have been performed in line 3, if for $c_k = b_i$, e_k contains b_j, then there is no interval $I_{j'}$ that overlaps with I_i and such that $b_j < b_{j'}$. Consequently, lines 7-10 correctly compute right(I_i). Since no read or write conflicts arise, the computation can be performed in the EREW-PRAM model. To address the complexity, we note that line 1 takes O(log n) time using O(n) processors if the sorting algorithm of Cole [3] is used. By (1.1)-(1.3) combined, lines 2-5 each takes O(log n) time using $\lceil n / \log n \rceil$ processors, and the conclusion follows. □

Corollary 1.1. The procedure Parallel_Find_Right is cost-optimal. □

Corollary 1.2. RIGHT can be solved sequentially in O(n log n) time. □

Procedure Parallel_Find_Next($\{I_i\}$);

```
{Input: a family of intervals $\{I_i \mid 1 \le i \le n\}$, with $I_i = [a_i, b_i]$.
 Output: next($I_i$) $(1 \le i \le n)$.
 0.  begin
 1.      let $c_1, c_2, ..., c_{2n}$ be the left and right-endpoins sorted in descending order†;
 2.      create $d_1, d_2, ..., d_{2n}$ with $d_i = \{b_j$,if $c_i$ is $a_j \mid\mid \infty$,otherwise} ;
 3.      perform prefix minimum on the sequence $d_1, d_2, ..., d_{2n}$, and
         let $e_1$ ,e2, ..., $e_{2n}$ be the result (i.e. $e_i = \min\{d_1, d_2, ..., d_i\}$);
 4.      let $I_{j_1}, I_{j_2}, ..., I_{j_{2n}}$ be the sequence such that $b_{j_i} = e_i$ $(1 \le i \le 2n)$;
 5.      for every $I_i$ $(1 \le i \le n)$ do in parallel begin
 6.          let k be the index of $b_i$ in the sequence $c_1, c_2, ..., c_{2n}$;
 7.          if $e_k \ne \infty$ then
 8.              next($I_i$) $\leftarrow I_{j_k}$
 9.          else
 10.             next($I_i$) $\leftarrow$ nil
 11.     end {for}
 12. end; {Parallel_Find_Next}
```

The following result which asserts that Parallel_Find_Next solves optimally the NEXT problem can be proved by an argument that mirrors that of Theorem 1 and is, therefore, omitted.

Theorem 2. Procedure Parallel_Find_Next correctly solves the NEXT problem in O(log n) time using O(n) processors in the EREW-PRAM model. □

Clearly, Theorem 2 and Lemma 6, combined, imply the optimality of Parallel_Find_Next. Furthermore, to obtain an optimal sequential algorithm to solve the same problem we again use the well-known trick of allowing a unique processor do sequentially the work of the O(n) processors, thus yielding an overall running time of O(n log n).

4. The Algorithms

Consider an interval graph G whose interval model is the family $I = \{I_i = [a_i, b_i] \mid a_i \le b_i, 1 \le i \le n\}$ of intervals on the real line. Our first algorithm computes an independent set in G having maximum cardinality over all the independent sets or, equivalently, a largest set of mutually non-overlapping intervals in the family I. To outline the algorithm, let F stand for the forest obtained by considering next(I_i) to be a pointer for all i = 1, 2, ..., n. Next, compute first(I) and then mark all the elements joining first(I) to the root of the tree containing first(I). Let L be the resulting list. Finally, record all the elements in L in the array S[1 .. n]. Clearly, the definition of "next" implies that all the entries in this linked list are mutually non-overlapping.

Procedure Maximum_Independent_Set($\{I_i\}$);

```
{Input: a family $I = \{I_i = [a_i, b_i] \mid a_i \le b_i, 1 \le i \le n\}$ of intervals;
 Output: a maximum independent set stored in the array S.}
 0. begin
 1.     compute first(I);
 2.     Parallel_Find_Next($\{I_i\}$);
```

---

† if $c_i = c_{i+1} = ...$, we assume that right-endpoints precede left-endpoints

```
3.  find the unique path L from first(I) to the root of the tree containing first(I);
4.  record all the elements in L in the array S;
5.  return(S)
6. end; {Maximum_Independent_Set}
```

Theorem 3. With an arbitrary family $I = \{I_i = [a_i, b_i] \mid a_i \leq b_i, 1 \leq i \leq n\}$ of intervals as input, procedure Maximum_Independent_Set correctly returns a maximum size independent set in $O(\log n)$ time using $O(n)$ processors in the EREW-PRAM model.

Proof. The correctness of the procedure follows from the following result proved in Golumbic [8]: there exists a maximum size independent set S in I such that first(I)$\in$ S and for every I_i that belongs to S, next(I_i) also belongs to S. To argue for the complexity, we note that computing first(I) takes $O(\log n)$ time and $O(n / \log n)$ processors; by Theorem 2, line 2 runs in $O(\log n)$ time using $O(n)$ processors; by Lemma 8 and Lemma 9 combined, line 3 takes $O(\log n)$ time and $O(n)$ processors; finally, by (1.2), line 4 takes $O(\log n)$ time and $O(n / \log n)$ processors. Since no read or write conflicts occur, the computation can be carried out in the EREW-PRAM model of computation. □

In [9] Gupta et al show that if an interval graph is given by its interval model, then the problem of computing a maximum size independent set in G has a lower bound of $\Omega(n \log n)$. Now Theorem 3 has the following consequence.

Corollary 3.1. Procedure Maximum_Independent_Set is cost-optimal. □

A *minimum clique cover* in a graph G is a minimum cardinality partition of the vertices of G into non-empty, disjoint sets such that each element of the partition is a clique. As it turns out, in the case of an interval graph, the cardinality of a minimum clique cover is equal to the cardinality of a maximum independent set. In addition, we can readily obtain a minimum clique cover once a maximum independent set is available (see [8] for details). Specifically, if $S=\{I_{k_1}, I_{k_2}, \ldots, I_{k_m}\}$ is a maximum size stable set in I, then there exists a minimum clique cover $C_1, C_2, \ldots, C_m$ of size *exactly* m, such that $I_{k_i} \in C_i$ for all $1 \leq i \leq m$ and for every interval $I_u \notin S$, I_u belongs to some clique C_j if, and only if, $b_{k_j} = \max\{b_{k_t} \mid I_{k_t} \in S$ and $a_u \leq b_{k_t} \leq b_u\}$.

It is easy to see that the problem of computing for every interval I_i, the subscript of the clique in the cover that contains I_i is an instance of the MULTIPLE SOURCE problem.

Procedure Minimum_Clique_Cover($\{I_i\}$);

{Input: a family $I = \{I_i = [a_i, b_i] \mid a_i \leq b_i, 1 \leq i \leq n\}$ of intervals;
Output: array E[1 .. n] containing for every I_i the subscript of the clique in the cover containing I_i.}

```
0. begin
1.    let $S=\{I_{k_1}, I_{k_2}, \ldots, I_{k_m}\}$ be returned by Maximum_Independent_Set($\{I_i\}$);
2.    let $c_1, c_2, \ldots, c_n$ be the right-endpoints in I sorted in ascending order;
3.    create labels $d_1, d_2, \ldots, d_n$ with $d_i = \{Ikj, \text{if } c_i \text{ is } b_{k_j} \mid\mid \text{nil}, \text{otherwise}\}$;
4.    perform prefix $\circ$ on $d_1, d_2, \ldots, d_n$ with $d_i \circ d_j = \{d_j, \text{if } d_j \neq \text{nil} \mid\mid d_i, \text{otherwise}\}$;
5.    let $e_1, e_2, \ldots, e_n$ be the result (i.e. $e_i = d_1 \circ d_2 \circ \cdots \circ d_i$ $1 \leq i \leq n$);
6.    for $1 \leq i \leq n$ do in parallel begin
7.       let $c_i = b_k$ for some k ($1 \leq k \leq n$);
8.       $E[k] \leftarrow e_i$
9.    end; {for}
10.   return(E)
11. end; {Minimum_Clique_Cover}
```

Theorem 4. With an arbitrary family $I = \{I_i = [a_i, b_i] \mid a_i \leq b_i, 1 \leq i \leq n\}$ of intervals as input, procedure Minimum_Clique_Cover correctly returns a clique cover of minimum size in $O(\log n)$ time using $O(n)$ processors in the EREW-PRAM model.

Proof. The correctness follows from the arguments in [8] along with Lemma 7. Since no read or write conflicts arise, the computation can be carried out in the EREW-PRAM model.

To address the complexity, note that by Theorem 3, line 1 takes $O(\log n)$ time using $O(n)$ processors; to sort in line 2, we use Cole's algorithm [3] running in $O(\log n)$ time using $O(n)$ processors. Further, by (1.3), lines 3-5 run in $O(\log n)$ time and use $O(n / \log n)$ processors. Finally, lines 6-8 take $O(1)$ time and $O(n)$ processors. □

Corollary 4.1. The procedure Minimum_Clique_Cover is cost-optimal.

Proof. The result can be obtained by showing that this result implies a solution to NON-OVERLAPPING INTERVALS. Since the lower bound for NON-OVERLAPPING INTERVALS is $\Omega(n \log n)$, every algorithm that correctly solves the minimum clique cover problem for a family of intervals must perform $\Omega(n \log n)$ operations in the worst case. Now the conclusion is implied by Theorem 4. □

Recall that a subset D of vertices of a graph is *dominating* if every vertex outside D is adjacent to some vertex in D. A dominating set is *minimum* if it has minimum cardinality among all dominating sets. The idea behind our parallel algorithm to compute a minimum dominating set is borrowed from the sequential algorithm of Booth and Johnson [2]: for every interval I_j define

$$\text{link}(I_j) = \{\text{right}(\text{next}(I_j)), \text{if right}(\text{next}(I_j)) \neq \text{nil} \mid\mid \text{next}(I_j), \text{otherwise}\}.$$

It is immediate that with this definition, I is organized as a forest F described by parent pointers. Let u stand for right(first(I)) or first(I) depending on whether right(first(I)) $\neq$ nil; next, we identify the unique path L joining u and the root of the tree containing u. Once L is constructed we record every element of L in an array D. The details are spelled out by the following procedure.

```
Procedure Minimum_Dominating_Set({$I_i$});
{Input: a family $I = \{I_i = [a_i, b_i] \mid a_i \leq b_i, 1 \leq i \leq n\}$ of intervals;
 Output: a minimum dominating set stored in the array D.}
 0. begin
 1.     Parallel_Find_Right({$I_i$});
 2.     Parallel_Find_Next({$I_i$});
 3.     u ← {right(first(I)), if right(first(I)) $\neq$ nil || first(I), otherwise} ;
 4.     construct link($I_j$) ($1 \leq j \leq n$);
 5.     mark all elements on the unique path L joining u to the root of the tree containing u;
 6.     record all the elements of L in the array D;
 7.     return(D)
 8. end; {Minimum_Dominating_Set}
```

Theorem 5. With an arbitrary family $I = \{I_i = [a_i, b_i] \mid a_i \leq b_i, 1 \leq i \leq n\}$ of intervals as input, procedure Minimum_Dominating_Set correctly returns a dominating set of the smallest size in $O(\log n)$ time using $O(n)$ processors in the EREW-PRAM model.

Proof. The correctness follows immediately from the correctness of the sequential algorithm in [2]. To see that the computation can be performed in the EREW-PRAM model, we only need show that line 4 can be implemented with no read or write conflicts. First, letting $\pi(i)$ be such that $I_{\pi(i)} = \text{next}(I_i)$, we sort the sequence $\pi(1), \pi(2), ..., \pi(n)$ in ascending order. Next, in the resulting sequence we mark every entry distinct from the preceding one (the first entry is automatically marked). Further, for every marked value $\pi(i)$, compute right($I_{\pi(i)}$). This value will be then transmitted to all the entries in the sorted list with the same value as $\pi(i)$. It is easy to see that this is an instance of the MULTIPLE SOURCE problem. Therefore, when this operation is done, every interval I_j knowns link(I_j). To address the complexity, note that by Theorem 1 and Theorem 2, lines 1 and 2 take $O(\log n)$ time and $O(n)$ processors. Line 3 amounts to computing the minimum of n entries (namely the right-endpoints) followed by one comparison. By the well-known binary fan-in technique this takes $O(\log n)$ time using $O(n / \log n)$ processors in the EREW-PRAM model. Line 4 runs in $O(\log n)$ time using $O(n)$ processors. To see that this is the case, note that sorting can be done in $O(\log n)$ time using $O(n)$ processors [3] and that by (1.1)-(1.3) all the operations described above run in $O(\log n)$ time using $O(n / \log n)$ processors. By Lemma 8 and Lemma 9, line 5 runs in $O(\log n)$ time using $O(n)$ processors. Finally, line 6 takes $O(\log n)$ time using $O(n / \log n)$ processors. The conclusion follows. □

Corollary 5.1. Procedure Minimum_Dominating_Set is cost-optimal.

Proof. As in Corollary 4.1, the algorithm implies a solution to NON-OVERLAPPING INTERVALS and is thus cost optimal. □

Consider a family $I = \{I_i = [a_i, b_i] \mid a_i \leq b_i, 1 \leq i \leq n\}$ of intervals on the real line. Define a linear order $<$ on I as follows: $I_i < I_j$ whenever $b_i < b_j$ or $b_i = b_j$ and $a_i \leq a_j$.

Let I_u and I_v be arbitrary intervals in I. Symmetry allows us to assume without loss of generality that $I_u < I_v$. Consider two intervals I_u and I_v in I with $I_u \lessdot I_v$. The idea is to construct a forest F by defining for every I_j in I

$$\text{link}(I_j) = \{\text{right}(I_j), \text{if } I_j \cap I_v = \varnothing \mid\mid I_v, \text{otherwise}\} .$$

We identify the unique path L joining I_u to I_v or to the root of some tree T of F. Once L is constructed we mark every element in L that is reachable from I_u along L. In case I_v is not marked (**nil**,-1) is returned, otherwise perform *list ranking* on L and record every marked element, along with its rank in L, in an array P. In particular, the *distance* $d(I_u,I_v)$ between I_u and I_v (that is, the length of the shortest path joining I_u and I_v) is also returned as a separate parameter. The details are spelled out by the following procedure.

```
Procedure Parallel_Shortest_Path($I_u$, $I_v$);
{Input: two intervals $I_u$ and $I_v$ with $I_u \lessdot I_v$;
 Output: P[1 .. rank($I_v$)] contains the intervals in a shortest path
         joining $I_u$ and $I_v$ if such a path exists, -1 is returned otherwise}
 0. begin
 1.    Parallel_Find_Right({$I_i$}); {assign one processor to every interval in I}
 2.    broadcast $I_v$ to all processors;
 3.    for all $I_i$ do in parallel
 4.       if $I_i \cap I_v \neq \varnothing$ then
 5.          link($I_i$) ← $I_v$
 6.       else
 7.          link($I_i$) ← right($I_i$);
 8.    find all the elements on the unique path L joining $I_u$ to $I_v$ or to the root of the tree
         containing u;
 9.    mark all elements in L;
10.    if $I_v$ is not marked then
11.       return(nil,-1); {there is no path joining $I_u$ and $I_v$}
12.    else begin
13.       perform list ranking on L;
14.       record all marked elements along with their rank in L in the array P;
15.       return(P,rank($I_v$)-1)
16.    end
17. end; {Parallel_Shortest_Path}
```

Theorem 6. Let G be an interval graph specified by its interval model $I = \{I_i \mid 1 \leq i \leq n\}$, and let I_u and I_v be arbitrary intervals in I such that $I_u \lessdot I_v$. procedure Parallel_Shortest_Path correctly returns a shortest path joining I_u and I_v in O(log n) time using O(n) processors in the EREW-PRAM model.

Proof. The proof of correctness is straight forward and is left to the reader. To address the complexity, note that by Theorem 1, line 1 runs in O(log n) time using O(n) processors; line 2 uses binary fan-in and runs in O(log n) time using O(n) processors; the for loop in lines 3-7 runs trivially in O(1) time using the n processors assigned to intervals. By Lemma 8 and Lemma 9 combined, line 8 takes O(log n) time using O(n) processors. Finally, by (1.3) lines 9 and 13 take O(log n) time using $\lceil n / \log n \rceil$ processors. Since no read or write conflict arises, the computation can be carried out in the EREW-PRAM model. □

By Lemma 3 and Theorem 6 combined we have the following result.

Corollary 6.1. Procedure Parallel_Shortest_Path is cost-optimal. □

For arbitrary intervals I_i, I_j in the family I, we let $d(I_i, I_j)$ stand for the shortest distance between I_i and I_j.

Lemma 10. diam(G) = d(first(I),last(I)). □

Lemma 11. For every I_i and every subscript k ($1 \leq k \leq n$), $d(I_i,I_k) \leq \max\{d(I_i,\text{first}(I)),d(I_i,\text{last}(I))\}$. □

Lemma 12. The center of a connected interval graph G specified by its interval model contains all the intervals I_i such that with δ standing for the diameter of G, $\max\{d(I_i,\text{first}(I)), d(I_i,\text{last}(I))\} = \lceil \delta / 2 \rceil$. □

Lemma 10, Lemma 11 and Lemma 12 combined, suggest the following approach to computing the center of an interval graph: compute the distance from first(I) to all the intervals in the family; by Lemma 10, the distance from first(I) to last(I) is exactly the diameter of G. Next, construct a new family I' of interval obtained from I by chaining every interval $I_i = [a_i, b_i]$ to $I'_i = [-b_i, -a_i]$. Again, compute the distance from first(I') to all the intervals in I': note that by our construction of I', this corresponds to the distance from last(I) to all the intervals in I. Finally, mark all the intervals satisfying the condition specified in Lemma 12. algorithm to compute the center of an interval graph (in case the center exists).

Procedure Parallel_Find_Center(I);

```
{Input: a family $I = \{I_i = [a_i, b_i] \mid a_i \le b_i, 1 \le i \le n\}$ of intervals on the real line;
 Output: the center of the corresponding interval graph if the center exists, $\varnothing$ otherwise.}
begin
    call Shortest_Path_first(I);
    {let $\delta$ be the distance between first(I) and last(I) and $d_{i_1}$ be the distances from first(I)
        to $I_i$ for $1 \le i \le n$}
    if $\delta = -1$ then return($\varnothing$);
    construct the family of intervals $I' = \{I_i = [-b_i, -a_i] \mid 1 \le i \le n\}$
    call Shortest_Path_first(I');
    {let $d_{i_2}$ be the distances from first(I') to $I_i'$ for $1 \le i \le n$}
    for each $I_i$ $(1 \le i \le n)$ do in parallel
        if $\max\{d_{i_1}, d_{i_2}\} = \lceil \delta / 2 \rceil$ then
            mark $I_i$;
    record all the marked intervals in the array C;
    return(C)
end; {Parallel_Find_Center}
```

Recall that as a result of running Parallel_Shortest_Path(first(I),last(I)), we get a shortest path $P = [\text{first}(I) = I_{i_0}, I_{i_1}, \ldots, I_{i_d} = \text{last}(I)]$ joining first(I) and last(I), along with the rank in P of every interval belonging to this path. It is easy to confirm that for every interval I_{i_k} in P, $\text{rank}(I_{i_k}) - 1$ is precisely the *distance* between first(I) and I_{i_k}. For an interval I_j which is not in P, find a subscript k such that both I_{i_k} and $I_{i_{k+1}}$ belong to P, and such that $I_j \cap I_{i_k} = \varnothing$, $I_j \cap I_{i_{k+1}} \neq \varnothing$. Now it is easy to confirm that the distance from first(I) to I_j is precisely $\text{rank}(I_{i_k}) + 1$. Note that the computation of the subscript k is an instance of the MULTIPLE SOURCE problem discussed in Section 3.

Procedure Parallel_Path_first(I);

```
{Input: a family $I = \{I_i = [a_i, b_i] \mid a_i \le b_i, 1 \le i \le n\}$ of intervals on the real line
 Output: the distance between first(I) and last(I) and the distance from first(I) to $I_i$ for
 $1 \le i \le n$ ;}
begin
    let $c_1, c_2, \ldots, c_{2n}$ be the left and right-endpoints sorted in ascending order$^{\dagger}$;
    compute first(I) and last(I);
    Parallel_Find_Right($\{I_i\}$);
    let $P = [\text{first}(I) = I_{i_0}, I_{i_1}, \ldots, I_{i_d} = \text{last}(I)]$ be returned by
        Parallel_Shortest_Path(first(I),last(I));
    let $b_{i_0}, b_{i_1}, \ldots, b_{i_d}$ be the corresponding sequence of right-endpoints;
    create $d_1, d_2, \ldots, d_{2n}$ with $d_i = \{\text{rank}(I_{i_k}), \text{if } c_i \text{ is } b_{i_k} \mid\mid 0, \text{otherwise}\}$ ;
    perform prefix $\circ$ on $d_1, d_2, \ldots, d_{2n}$ with $d_i \circ d_j = \{d_j, d_j \neq 0 \mid\mid d_i, \text{otherwise}\}$ ;
    let $e_1, e_2, \ldots, e_{2n}$ be the result (i.e. $e_i = d_1 \circ d_2 \circ \cdots \circ d_i$ $1 \le i \le n$);
    for every $I_i$ $(1 \le i \le n)$ do in parallel
        if $I_i$ in P then
            $d_{i_1} \leftarrow \text{rank}(I_i) - 1$;
        else
            {assume the index of $a_i$ in the sequence $c_1, c_2, \ldots, c_{2n}$ is k}
```

---

† if $c_i = c_{i+1} = \ldots$, we assume that left-endpoints precede right-endpoints

```
13.         $d_{i_1} \leftarrow e_k+1$;
14.     return(d, $d_{i_1}$ for $1 \le i \le n$).
15. end; {Parallel_Path_first}
```

Theorem 7. Let G be an interval graph specified by its interval model $I = \{I_i \mid 1 \le i \le n\}$. Procedure Parallel_Path_first correctly computes the distance from First(I) to all the intervals in I in O(log n) time using O(n) processors in the EREW-PRAM model.

Proof. To prove the correctness, we note that when line 4 is completed, we have a shortest path $P = [\text{first}(I)=I_{i_0}, I_{i_1}, ..., I_{i_d}=\text{last}(I)]$ joining first(I) and last(I). Further, let the corresponding sequence of right-endpoints be $b_{i_0}, b_{i_1}, ..., b_{i_d}$. Consider an interval I_j which does not belong to P. We want to identify the first interval I_{i_p} in P to the left of I_j which is disjoint from I_j. Clearly, the shortest path from first(I) to I_j is first(I) = $I_{i_0}, I_{i_1}, ..., I_{i_p}, I_{i_{p+1}}, I_j$ implying that the distance from first(I) to I_j is the rank(I_{i_p})+1. Now the correctness follows from Lemma 7. To address the complexity, note that line 1 runs in O(log n) time using O(n) processors using Cole's sorting algorithm [3], line 2 runs in O(log n) time using $\lceil n / \log n \rceil$ processors; by Theorem 1 line 3 takes O(log n) time using O(n) processors; by Theorem 6, line 4 runs in O(log n) time using O(n) processors. Next, by (1.1)-(1.3), lines 5-8 run in O(log n) time using $\lceil n / \log n \rceil$ processors. Finally, lines 9-13 run in O(1) time using O(n) processors. Since no read or write conflicts arise the computation can be performed in the EREW-PRAM model. □

Theorem 7 implies the following result arguing about the correctness and running time of our parallel algorithm to compute the center of an interval graph specified by its interval model.

Theorem 8. Let G be an interval graph specified by its interval model $I = \{I_i \mid 1 \le i \le n\}$. Procedure Parallel_Find_Center correctly computes the center of G in O(log n) time using O(n) processors in the EREW-PRAM model. □

Corollary 8.1. Procedure Parallel_Find_Center is cost optimal. □

References

1. A. A. Bertossi and M. A. Bonuccelli, Some parallel algorithms on interval graphs, *Discrete Applied Mathematics* (1987) 101-111.
2. K. Booth and J. H. Johnson, Dominating sets in chordal graphs, *SIAM Journal on Computing*, 11 (1982) 191-199.
3. R. Cole, Parallel Merge Sort, *SIAM Journal on Computing*, 17 (1988) 770-785.
4. R. Cole and U. Vishkin, Approximate parallel scheduling. Part I: The basic technique with applications to optimal parallel list ranking in logarithmic time, *SIAM Journal on Computing*, 17, (1988) 128-142.
5. E. Dekel and S. Sahni, Binary trees and parallel scheduling algorithms, *IEEE Trans. Comput.* 32 (1983) 307-315.
6. E. Dekel and S. Sahni, Parallel Scheduling Algorithms, *Operations Research*, 31 (1983) 24-49.
7. P. C. Gilmore and A. J. Hoffman, A characterization of comparability graphs and interval graphs, *Canad. J. of Math.* 16, (1964) 539-548.
8. M. C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.
9. U. I. Gupta, D. T. Lee, and J. Y. T. Leung, Efficient algorithms for interval graphs and circular-arc graphs, *Networks*, Vol. 12 (1982) 459-467.
10. C. P. Kruskal, L. Rudolph, and M. Snir, Efficient parallel algorithms for graph problems, *Algorithmica*, 5 (1990) 43-64.
11. F. P. Preparata and M. I. Shamos, Computational geometry, Springer-Verlag, New York, 1988.
12. R. E. Tarjan and U. Vishkin, An efficient parallel biconnectivity algorithm, *SIAM Journal on Computing*, 14 (1985) 862-874.

Chord-free Path problems on Interval Graphs
(abbreviated version)

Glenn K. Manacher
Department of Mathematics,
Statistics, and Computer Science
University of Illinois at Chicago
Chicago, ILL 60680 USA

and

Srinivasa Rao Arikati
C. Pandu Rangan
Department of Computer Science
and Engineering
Indian Institute of Technology
Madras 600 036, India

ABSTRACT. Define a **chord-free path** (CFP) between two given vertices a, b of a graph $G = (V, \widehat{E})$ to be a path $p_1 p_2 \ldots p_k$, $a = p_1$, $b = p_k$, such that no two nonadjacent vertices on the path are connected by an edge. The **parity path problem**, (PPP) posed by Hsu [Hs 87], asks whether there exists an even - (or, alternatively, odd) - length CFP. The problem for general graphs is not known to be in P or to be NP-complete.

In [AMR 90] we showed that the PPP can be solved in time $O(n + m)$ for interval graphs, where $m = \#\widehat{E}$.

For interval graphs, with the representation provided, we show that the PPP may be solved in time $O(n \log n)$ and space $O(n)$, where $n = \#V$. We then pose a generalization, the **semigroup path problem** (SPP), in which each vertex ν_i is given a weight w_i, where $w_i \in \sum$ and $\sum$ is a finite semigroup. The problem, given two intervals a and b is to determine, for each $\sigma \in \sum$, is whether there exists a CFP $p_1 p_2 \ldots p_k$, $p_1 = a$, $p_k = b$, such that the semigroup product $w_{p_1} w_{p_2} \ldots w_{p_k}$ is equal to σ.

We show that this may be decided, and a CFP exhibited for each $\sigma \in \sum$ if one exists, in time $O(|\sum| \, n \log n)$ and space $O(|\sum| \, n)$. □

The input consists of a list of $2n$ symbols $LE(i)$ and $RE(i)$ for $1 \leq i \leq n$, where LE and RE denote resp. the left and right endpoints of interval i. Intervals are labelled by their LE's in a left-to-right scan.

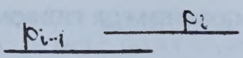
A moment's reflection shows that any CFP must be simple. Assume wlog that $a \neq b$, $LE(a) < LE(b)$, $ab \notin \widehat{E}$, and $\#p \geq 3$ for a CFP p from a to b. [AP 89] established that for all c neither a nor b in p, $RE(c) > RE(a)$ and $LE(c) < LE(b)$. Denote the interval set by I. We first preprocess I by removing all c violating these inequalities. Call the resulting set XI and let $n = \#XI$ in the sequel. Now complete the preprocessing by extending $LE(a)$ to the extreme left in XI and $RE(b)$ to the extreme right. Let $\tau \in (-1, +1)$ be a *parity symbol* denoting resp. odd and even parity. Note that in XI, a is XI_1 and b is XI_n.

Lemma 1. Any CFP in I from a to b is present without change (comprises the same sequence of intervals) in XI. Further, using [AP 89], we have

Corollary 1. Any CFP $p(a,b)$ in XI satisfies

a) For $1 < i < m$, p_i intersects only p_{i-1} and p_{i+1};

b) For $1 < i \leq m$, p_i right-overlaps p_{i-1}. □



The algorithm uses the following definitions for all c in XI.

Definition 1. Let $p_\tau(c) = q_1 \ldots q_k$, $q_1 = c$, $q_k = b$ be a parity τ CFP from c to b in XI whose *second* element (following c) has the largest possible label, if such a path exists at all. Define $B_\tau(c) = q_2$ if $p_\tau(c)$ exists, else $B_\tau(c) \equiv 0$. If $c = b$, then define $B_{-1}(b) \equiv n + 1$ (a dummy value) and $B_{+1}(b) \equiv 0$.

Theorem 1. Algorithm "compute $B_\tau(a)$" will compute $B_\tau(e)$ for all $e \in XI$ and all $\tau \in (-1, +1)$. Moreover the transitive closure of $B_\tau(c)$, with τ *alternating*, (the closure ending with b), is a CFP from c to b of parity τ if such a path exists.

Algorithm Compute $B_\tau(a)$

Let $r_1 r_2 \ldots r_{\#XI}$ denote the sequence of intervals in XI whose *right* endpoints are encountered in a right-to-left scan of XI.

Let S_τ denote two subsets of intervals of XI, not necessarily disjoint.

Note that $b = r_1$ and $a = r_{\#XI}$.

```
1. --initialization--   $(\forall\tau)\ S_\tau := \emptyset$;
        $(\forall e \in XI)\ (\forall\tau) B_\tau(e) := 0$;
        $B_{-1}(b) := \#XI + 1$;
            -- $S_\tau$ is the set of intervals thus far scanned for which $B_\tau(e) \neq 0$ --
2. For $k := 2$ to $\#XI$ do
     For $\tau := +1, -1$ do
3.     $U_\tau := \{s \in S_{-\tau} \mid s > r_k$ and $s < RE(r_k)$ and
         $B_{-\tau}(s) \gg r_k\}$; -- remember, $r_k$ is a shorthand for $LE(r_k)$ --
4.     If $U_\tau \neq \emptyset$ then $B_\tau(r_k) := \max(u \mid u \in U_\tau)$;
         $S_\tau \leftarrow S_\tau \cup \{r_k\}$
       Fi
     Od
   Od
5. For $\tau := +1, -1$ do
     If $B_\tau(a) \neq 0$ then write ("A parity" $\tau$ "path is" $BP_\tau(a)$)
       -- the alternating $\tau$ backpointer closure from $a$ to $b$ --
     Fi
   Od
End Compute $B_\tau(a)$.
```

The remainder of the paper consists of proving the space and time behavior of the algorithm, particularly with regard to [McC 85], and also the generalization to the semigroup path problem. The full paper is available.

REFERENCES

[AP 89] Arikati, S. R. and Pandu Rangan, C. Linear algorithms for parity path problems on circular arc graphs, in Workshop on Algorithms and Data Structures, Ottawa, Canada, 1989.

[Hs 87] Hsu, W.-L. Recognizing Perfect Planar Graphs. *J. ACM*, **34**, 255-258, 1987.

McC 85 McCreight, E. M. Priority Search Trees. *SIAM J. Comp.*, **14**, 2, 1985, 257-275.

AN *NC* ALGORITHM FOR THE DOMINATION IN THE PERMUTATION GRAPHS

C. Rhee
Department of Mathematics and Computer Science
Eastern Kentucky University, Richmond, KY 40475

S. K. Dhall and S. Lakshmivarahan
School of Electrical Engineering and Computer Science
University of Oklahoma, Norman, Oklahoma 73019

1. INTRODUCTION

Let $G = (V, E)$ be an undirected graph and let D_1 and D_2 be subsets of V. Then D_1 is said to *dominate* D_2 if each vertex in D_2 is adjacent to some vertex in D_1. A subset $D \subseteq V$ is called a *dominating* set for G if D dominates $V - D$. A graph G is called a *weighted* graph if there exists a weight function $w : V \rightarrow R$ (the set of reals), where $w(v)$ is called the weight of vertex v. In this paper, we assume that all vertices have weight greater than 0. Let $W(D) = \sum_{v \in D} w(v)$. A dominating set $D^* \subseteq V$ is called a *minimum weight* dominating set for G if $W(D^*) = \min\{W(D) \mid D \subseteq V$ and D is a dominating set for $G.\}$. Let $< n > = \{1, 2, ..., n\}$ and $\Pi = (\Pi(1)\ \Pi(2) \cdots \Pi(n))$ be a permutation on $< n >$. For a given permutation Π, we construct an undirected graph $G(\Pi) = (< n >, E)$, where $i\ j \in E$ if and only if $(i - j)(\Pi^{-1}(i) - \Pi^{-1}(j)) < 0$. Any undirected graph G is called a *permutation* graph if there exists a Π such that G is isomorphic to $G(\Pi)$, i.e. $G \approx G(\Pi)$. The permutation graph $G = (V, E)$ will be denoted as $G = (\Pi(1)\Pi(2) \cdots \Pi(n))$ in this paper, where Π is the underlying permutation for G. Farber and Keil [1] developed $O(n^3)$ sequential algorithms for finding a minimum weight dominating set and a maximum weight independent set in permutation graphs. They also described an $O(n^2)$ algorithm for the minimum cardinality dominating set problem. Atallah and Kosaraju [2] described an $O(n \log n)$ algorithm for finding a minimum weight independent dominating set, which improved one of the Farber and Keil's results. In this paper we present an *NC* algorithm for finding a minimum weight dominating set in permutation graphs. This result is interesting since there exists no known *NC* algorithm for this problem and the current sequential algorithms do not directly imply that they can be easily parallelized.

2. THE ALGORITHM

Let $G = (V, E)$ with $V = \{1, 2, ..., n\}$ be a weighted permutation graph, where each node i has a weight $w(i)$. We construct a weighted digraph $H = (V^H, E^H)$ from $G = (V, E)$ in the following way: Augment V with two additional vertices 0 and $n+1$ such that $\Pi(0) = 0$, $\Pi(n+1) = n+1$, and $w(0) = w(n+1) = 0$, denoting the augmented vertex set as V^+. Let $G^+ = (V^+, E)$ be the resulting graph. First, to construct V^H from V^+, let $V^H = \{v_{ij} \mid i, j \in V^+, i \leq j$, and either $i = j$ or $i\ j \in E\}$. Intuitively, each vertex in V^H corresponds to either a vertex in V^+ or a pair of vertices in V^+ which are adjacent to each other. In the sequel, we use the following definitions: Let $v_{ij} \in V^H$ and $k \in V^+$. Then k is said to be *embedded* in v_{ij} if either $k = i$ or $k = j$. $S(i_1, i_2, ..., i_k)$ denotes the set of vertices $i_1, i_2, ..., i_k$ in V^+. For vertices $v_{ij}, v_{i'j'} \in V^H$, define the following relations: (i) $v_{ij} \approx v_{i'j'}$ if (a) $v_{ij} \neq v_{i'j'}$, and (b) there are $k \in S(i, j)$ and $k' \in S(i', j')$ such that k and k' are adjacent to each other in G^+. (ii) $v_{ij} \leq v_{i'j'}$ if (a) $i \neq j$ and $i' \neq j'$, (b) $v_{ij} \approx v_{i'j'}$, (c) for each $k \in \{i, j\}$, there is $k' \in \{i', j'\}$ such that either ($k < k'$ and $\Pi^{-1}(k) < \Pi^{-1}(k')$) or ($k = k'$), and (d) for each $k' \in \{i', j'\}$, there is $k \in \{i, j\}$ such that either ($k < k'$ and $\Pi^{-1}(k) < \Pi^{-1}(k')$) or ($k = k'$). (Note that this relation is defined only between vertices of V^H which correspond to the edges in G^+.) (iii) $v_{ij} < v_{i'j'}$ if $k < k'$ and $\Pi^{-1}(k) < \Pi^{-1}(k')$ for each $k \in S(i, j)$ and each $k' \in S(i', j')$.

Now, we construct the edge set E^H as follows. For each pair of vertices $v_{ij}, v_{i'j'} \in V^H$, $(v_{ij}, v_{i'j'}) \in E^H$ with weight $\sum_{s \in S(i,j) - S(i',j')} w(s)$ iff either (i) $v_{ij} \leq v_{i'j'}$ or (ii) $v_{ij} < v_{i'j'}$ and there is no vertex $k \in V^+$ such that $v_{ij} < v_{kk} < v_{i'j'}$. As we shall see later, if any vertex i in V^+ is embedded in more than one vertex of V^H on a path in H, then $w(i)$ is counted only once in determining the length of the path. Let $P = v_{i_0j_0} v_{i_1j_1} \cdots v_{i_kj_k} v_{i_{k+1}j_{k+1}}$ be a path from v_{00} to $v_{(n+1)(n+1)}$ in H with $v_{i_0j_0} = v_{00}$ and $v_{i_{k+1}j_{k+1}} = v_{(n+1)(n+1)}$. Then $|P|$ denotes the length of P, i.e. $|P| = \sum_{s=0}^{k} w(v_{i_sj_s}, v_{i_{s+1}j_{s+1}})$, the summation of weight of each edge in P. In this paper a path P in H indicates a directed path from $v_{i_0j_0} (= v_{00})$ to $v_{i_{k+1}j_{k+1}} (= v_{(n+1)(n+1)})$, unless otherwise stated. For a path P in H, $V_G(P)$ indicates the set of vertices i in V^+ such that i is embedded in some vertex on P. Let $V' \subseteq V^+$. Then $W(V')$ denotes the weight of V', which is the sum of weights of vertices in V'. A subset V' of V^+ is said to be *k-clique bounded* if the cardinality of every clique in the subgraph induced by V', denoted by $G_{V'}$, is at most k. Again, let V' be a subset of V^+ and P be a path in H. Then P is said to *correspond* to V' (or equivalently V' is said to *correspond* to P) if $V' = V_G(P)$ and $|P| = W(V')$.

OBSERVATION 1: Let $v_{i_t j_t}$ and $v_{i_r j_r}$, $t < r$, be vertices on a path $P = v_{i_0 j_0} v_{i_1 j_1} \cdots v_{i_k j_k} v_{i_{k+1} j_{k+1}}$ in H. Then, (i) either $v_{i_t j_t} \leq v_{i_r j_r}$ or $v_{i_r j_r} < v_{i_t j_t}$. (ii) If $v_{i_t j_t} \leq v_{i_r j_r}$, then $v_{i_t j_t} \leq v_{i_s j_s}$ for each s, $t+1 \leq s \leq r$, and $v_{i_s j_s} \leq v_{i_r j_r}$ for each s, $t \leq s \leq r-1$. (iii) The subgraph induced by $S(i_t, j_t, i_r, j_r)$ does not contain a clique of size greater than 2.

OBSERVATION 2: The digraph H is acyclic.

LEMMA 1. For any path P in H, there is a 2-clique bounded dominating set for G^+ which corresponds to P, and vice versa.

LEMMA 2. Any minimum weight dominating set for G^+ is 2-clique bounded.

PROOF: Let D be a minimum weight dominating set for G^+. Suppose that $D' \subseteq D$ be a subset such that the induced subgraph $G^+_{D'}$ is a k-clique with $k > 2$, i.e. $G^+_{D'}$ is a subpermutation $(i_1 i_2 \cdots i_k)$ satisfying $i_1 > i_2 > \cdots > i_k$ and $\Pi^{-1}(i_1) < \Pi^{-1}(i_2) < \cdots < \Pi^{-1}(i_k)$. Let j be a vertex in V which is adjacent to i_s for some $s \in \{2, 3, ..., k-1\}$. (If no vertex is adjacent to such i_s, then i_s could be removed from D and the resulting set $D - \{i_s\}$ would be still a dominating set for G^+.) Recall that the weight of each node is positive. Then clearly j is adjacent to either i_1 or i_k. This implies that D can be replaced by $\{i_1, i_k\}$ and the resulting set is still a dominating set for G^+, which is a contradiction. Therefore, any minimum weight dominating set for G^+ is 2-clique bounded. □

THEOREM 1. Any shortest directed path from v_{00} to $v_{(n+1)(n+1)}$ in H corresponds to a minimum weight dominating set for G^+.

THEOREM 2. Finding a minimum weight dominating set for a permutation graph can be done in $O(\log n)$ time with polynomially many processors on the CRCW model.

PROOF: The acyclic digraph H can be constructed in $O(\log n)$ time using $O((m+n)^2)$ processors on the CREW model since $|V^H| \leq m+n$ and finding a shortest path in H can be done in $O(\log n)$ time with $O((m+n)^{3+\epsilon})$ processors on the CRCW model [3], where m denotes the number of edges in G. □

REFERENCES

1. M. Farber and J. M. Keil, "Domination in Permutation Graphs", J. of Algorithms, Vol 6, pp 309 - 321, 1985.
2. M. J. Atallah and S. R. Kosaraju, "An Efficient Algorithm for Maxdominance, with Applications", Technical Rep., Purdue Univ.
3. R. C. Paige and C. P. Kruskal, "Parallel Algorithms for Shortest Path Problems", 26th IEEE FOCS, pp 14 - 20, 1985.

Ameliorating Sequences of DNC Algorithms for Separators and Breadth-First Search in Planar Graphs

Fang Wan
Department of Computer Science
Indiana University
Bloomington, Indiana 47405

Abstract

An *ameliorating sequence* of algorithms for a problem is a sequence of algorithms that use previous algorithms from the sequence to improve with respect to some criterions (e.g., time, processors, or output quality). In this paper, we improve inefficient initial parallel algorithms for the problems of finding $O(\sqrt{n})$ size cyclic separators (Sep) and breadth-first search trees (BFS) in planar graphs in terms of processor-time products by alternately producing new algorithms for each problem from the previous algorithms for the other problem. The processor complexities of our Sep algorithms and BFS algorithms for planar graphs converge toward $n^{1.5}$, and the sizes of the separators found by our linear-processor small separator algorithms converge toward $n^{2/3}$. Previous algorithms for these problems either use significantly more processors, are randomized, or find notably larger separators. All of our algorithms are deterministic on the parallel random access machine (PRAM) and use poly-log time and polynomial processors in n, the sizes of input graphs.

1 Introduction

An $s(n)$-cycle separator of a planar graph of n vertices is a simple cycle which has no more than $O(s(n))$ vertices and contains no more than $2n/3$ vertices either inside or outside. Cycle separators are also vertex separators and $\sqrt{dn}$-cycle separators of undirected planar graphs are optimal within a constant factor in the worst case, where d is the maximum face size [10,11]. Any planar graph can be triangulated and treated as undirected to guarantee the existence of $\sqrt{n}$-cycle separators which imply $\sqrt{n}$ size vertex separators. The $\sqrt{dn}$-cycle separator (Sep) problem and related problems have been well studied both in serial computation models and in parallel computation models [10,3,11,6,4,7,5,17]. Breadth-first search (BFS) is a fundamental problem in computational graph theory [1,9,14,16]. Parallel $\sqrt{dn}$-cycle separator algorithms have proven to be important tools to develop parallel BFS algorithms [13]. In this paper, a sub-linear size separator is referred as a *small* separator and, without further mention, all BFS algorithms and $\sqrt{dn}$-cycle separator algorithms are for planar graphs or undirected planar graphs respectively. All computation complexities are based on the concurrent-read concurrent-write parallel random access machine (CRCW PRAM) [8].

In contrast with optimal linear-time sequential algorithms for the BFS problem and the Sep problem [1,10,11], previous parallel algorithms for the two problems either use too many processors, depend on randomized techniques, or find large separators [11,6,7,13,17]. Much progress has to be made before linear-processor *deterministic polylog time* (DNC) algorithms for the two problems are found to achieve near-maximum speed up and near-optimal processor-time products. In this paper, we present a sequence of DNC BFS algorithms and DNC $\sqrt{dn}$-cycle separator algorithms. These algorithms use at most $O(\log^{i+2} n)$ time and $n^{1.5+4\epsilon^i}$ processors for any fixed i, where $\epsilon \leq 1/2$. Our results use significantly fewer processors or find notably smaller separators.

Our algorithms are based on new techniques for solving the following two problems: finding a BFS tree on a face incidence graph (usually not a planar graph) by a BFS algorithm for planar graphs and using DNC BFS algorithms and DNC Sep algorithms to alternatively ameliorate each other in terms of processor-time products.

The bottle neck in Miller's original $\sqrt{dn}$-cycle separator algorithm is performing a BFS on the face incidence graph of an undirected planar graph to find complete face layers of the undirected planar graph [11]. Since a face incidence graph usually is not a planar graph, parallel BFS algorithms for general graphs have been used and the best such algorithm uses $n^{2.376}$ processors [7]. However, careful observations reveal that BFS on face incidence graphs is actually as easy as BFS on planar graphs. We construct a graph called *face distance graph* for a face incidence graph by adding a vertex in each face of the original graph of the face incidence graph, connecting a new vertex to all vertices on the boundary of the corresponding face, and deleting all edges of the original graph. We prove that the distance between two vertices of a face incidence graph is exactly one half of the distance between the corresponding vertices in the corresponding face distance graph and, the best of all, the face distance graph is planar! Using the new tool, we reduce the Sep problem to BFS on planar graphs, in contrast with reducing the Sep problem to BFS on general graphs in previous works. Thus we provide a new prospect, realized later in the paper, of improving BFS algorithms and Sep algorithms. Our new approach to BFS on face incidence graphs also improves parallel implementations of algorithms using complete face layers of planar graphs, e.g., Baker's approximation algorithms for some NP-complete problems on planar graphs [2].

Although BFS does not use small separators in serial computations, an efficient DNC Sep algorithm is the only known tool to realize a DNC BFS algorithm using fewer processors than DNC matrix multiplication algorithms. Pan and Reif constructed a DNC BFS algorithm using $s^3(n)$ processors when a family of $s(n)$ size separators is given [13]. Using their construction, Gazit and Miller's randomized algorithm implies a randomized BFS algorithm using $n^{1.5}$ processors. Given a DNC algorithm for BFS on face incidence graphs using n^{β} processors, Wan constructed a DNC BFS algorithm using $n^{\beta'}$ processors by finding $O(n^{\beta'/3})$ size separators with $n^{\beta'}$ processors, where $1.5 < \beta$ and $\beta' < \beta$ [17]. Since the new BFS algorithm for planar graphs can do BFS on face incidence graphs, we can develop another BFS algorithm using $n^{\beta''}$ processors, where $\beta'' < \beta'$. Repeating this process, we get an ameliorating sequence of BFS algorithms and $\sqrt{dn}$-cycle separator algorithms.

Specifically, our major result is an ameliorating sequence of BFS algorithms and $\sqrt{dn}$-cycle Sep algorithms. If the ith BFS algorithm uses $O(\log^{\alpha_i} n)$ time and $n^{1.5+\epsilon_i}$ pro-

Previous Results	$\sqrt{dn}$-Cycle Separator		Single-Source Shortest Paths		All-Pairs Shortest Paths	
	time	processor	time	processor	time	processor
DNC[11,7,8,13,17]	$\log^2 n$	$n + f^M$	$\log^2 n$	$n^{1.84}$	$\log n$	n^3
RNC[6,13]	$\log^2 n$	$n + f^{1+\epsilon}$	$\log^3 n$	$n^{1.5}/\log^2 n$	$\log^3 n$	$n^2/\log^2 n$

Our Results	$\sqrt{dn}$-Cycle Separator		Single-Source Shortest Paths		All-Pairs Shortest Paths	
DNC	$\log^{i+1} n$	$n^{1.5+\epsilon_i}$	$\log^{i+2} n$	$n^{1.5+\epsilon_i}$	$\log^4 n$	$n^2/\log^2 n$
i=1	$\log^2 n$	$n^{1.839}$	$\log^3 n$	$n^{1.839}$		
i=5	$\log^6 n$	$n^{1.509}$	$\log^7 n$	$n^{1.509}$		

Notes	Finding Sublinear Cyclic Separators on n Processors		
	time	processor	cycle size
Previous DNC[17]	$\log^2 n$	n	$max(\sqrt{dn}, n^{0.79})$
Ours	$\log^{i+1} n$	n	$max(\sqrt{dn}, n^{(2+2\epsilon_i)/(3+2\epsilon_i)})$
i=1	$\log^2 n$	n	$max(\sqrt{dn}, n^{0.79})$
i=5	$\log^6 n$	n	$max(\sqrt{dn}, n^{0.674})$

* All problems are restricted on planar graphs or undirected planar graphs accordingly.

** $\epsilon_{i+1} = 3\epsilon_i/(6+2\epsilon_i)$ and $\epsilon_0 = 0.876$.

Figure 1: A summary

cessors, the $(i+1)$th $O(\sqrt{n})$ size Sep algorithm uses $O(\log^{\alpha_i} n)$ time and $n^{1.5+3\epsilon_i/(2\epsilon_i+6)}$ processors, the $(i+1)$th linear-processor cycle separator algorithm uses $O(\log^{\alpha_i} n)$ time to find a $max(\sqrt{dn}, n^{(2+2\epsilon_i)/(3+2\epsilon_i)})$-cycle separator, and the $(i+1)$th BFS algorithm uses $O(\log^{\alpha_i+1} n)$ time and $n^{1.5+3\epsilon_i/(2\epsilon_i+6)}$ processors, where i is bound by any constant, α_0=2, and ϵ_0=0.876. A sub-linear factor decrease in the processor complexity of Sep algorithms or BFS algorithms requires a logarithm factor increase in their time complexity. The processor complexities converge to $n^{1.5}$. Furthermore, if our ameliorating sequence uses a BFS algorithm that requires $s^\gamma(n)$ processors when $s(n)$ size separators are given, its processor complexities converge to $s^{\gamma/2}(n)$, where $2 \leq \gamma \leq 3$. Using Pan and Reif's reduction, we also construct an all-pairs shortest paths algorithm using $O(\log^4 n)$ time and $n^2/\log^2 n$ processors for planar graphs.

Ours and previous results are summarized in the Figure 1. In Section 2, we show how to perform BFS on a face incidence graph using a BFS algorithm for planar graphs with the help of a corresponding face distance graph. In Section 3, we introduce an ameliorating sequence of BFS algorithms and Sep algorithms. Planar embedding is assumed through out the paper [15].

2 BFS On Face Incidence Graphs

In Miller's original $\sqrt{dn}$-cycle separator algorithm, complete face layers surrounding a center face are essential to find an area, like a piece of pie, whose boundary is a $\sqrt{dn}$-cycle separator. The complete face layers are calculated by finding a BFS tree on the face incidence graph of the original graph. Except for this part of calculations, the algorithm is efficient and uses $O(\log n)$ time and n processors. The face incidence graph of a planar graph G is an undirected graph which has a distinct vertex for each face of

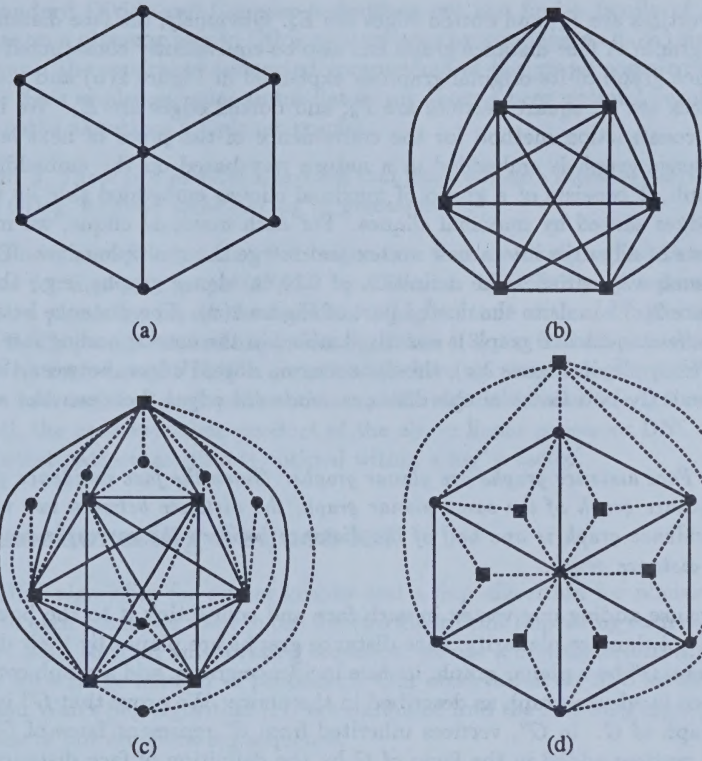


Figure 2: (a) A planar graph, G. (b) The face incidence graph, G', of G. (c) The face distance graph of G is represented by all vertices and dotted edges and is constructed from G' represented only by square vertices and solid edges. (d) The face distance graph of G is represented by all vertices and dotted edges and is constructed directly from G represented only by round vertices and solid edges.

G and an edge between two vertices if and only if their corresponding faces in G share at least a vertex. The face incidence graph of a planar graph usually is not a planar graph any more. For example, Figure 2(a) is a planar graph and its face incidence graph, Figure 2(b), is not a planar graph. Since current parallel BFS algorithms for general graphs have to use matrix multiplication and $O(n^2)$ is a trivial lower bound of the processor-time complexity of any DNC matrix multiplication algorithm, we have to find a way to get around this bottle neck to achieve a small cycle separator algorithm using fewer processors.

We show that finding a BFS tree on the face incidence graph of a planar graph is as easy as finding a BFS tree on a planar graph even when the face incidence graph is not planar. Our tool is a type of graphs called *face distance graphs.* The face distance graph of a planar graph G=(V,E) is defined as G_d=($V \cup F_d$, E_d), where F_d is a set of vertices in the centers of the faces of G and E_d=$\{(v, f) \mid v \in V,\ f \in F_d,$ *and v is on the boundary of f in G*$\}$. A face distance graph can be constructed directly from its original graph by its definition as explained in Figure 2(a) and Figure 2(d), where round vertices are

V, square vertices are F_d, and dotted edges are E_d. Obviously, the face distance graph is a planar graph. A face distance graph can also be equivalently constructed from the face incidence graph of its original graph as explained in Figure 2(b) and 2(c), where round vertices are V, square vertices are F_d, and dotted edges are E_d. We introduce the second construction method for the convenience of the proof of next lemma. If a face incidence graph is embedded in a nature way based on the embedding of its original graph, it consists of a group of maximal cliques embedded side by side. We duplicate edges shared by maximal cliques. For each maximal clique, we merge the middle points of all edges into a new vertex and merge the multiple edges. The graph formed in such way satisfies the definition of face incidence graphs, e.g., the dotted part of Figure 2(c) equals to the dotted part of Figure 2(d). The distance between two vertices in a face incidence graph is exactly doubled in the corresponding face distance graph. For example, in Figure 2(c), the distance, via dotted edges, between two square vertices is exactly two times of the distance, via solid edges, between the same two vertices.

Lemma 1 *Face distance graphs are planar graphs. Given the face incidence graph and the face distance graph of the same planar graph, the distance between two vertices of the face incidence graph is one half of the distance between the corresponding vertices of the face distance graph.*

Proof: Because adding one vertex in each face and connecting it to the boundary of the face do not damage planarity, Face distance graphs are planar by their definition. Let G, G', and G'' be a planar graph, its face incidence graph, and a graph constructed from the face incidence graph as described in the above. We prove that G'' is the face distance graph of G. In G'', vertices inherited from G' represent faces of G and are mapped to vertices added in the faces of G by the definition of face distance graphs. Maximal cliques of G' represent vertices of G and vertices formed by merging middle points of edges of maximal cliques are mapped to the vertices inherited from G in G''. So there is an one-to-one mapping between vertices of G'' and vertices of the face distance graph of G. Furthermore, a vertex of G' is in a maximal clique if and only if the original vertex represented by the clique is on the boundary of the original face represented by the vertex, so a vertex representing an original face is connected and is only connected to vertices on the boundary of the original face.

The rest of the proof for the second part of the lemma is based on the simple fact that the distance between two vertices in the face incidence graph, G', is 1 if and only if the distance between the corresponding vertices in the corresponding face distance graph, G'', is 2. According to the construction of a face distance graph from a face incidence graph, the fact is true because every edge in a face incidence graph is replaced by two concatenate edges in the corresponding face distance graph. ∎

Theorem 1 *If there is a parallel algorithm that performs BFS on a planar graph in* $O(\log^\alpha n)$ *time using* n^β *processors, a* $\sqrt{dn}$*-cycle separator on an undirected planar graph can be found in* $O(\log^\alpha n)$ *time with* n^β *processors, where* $\alpha,\beta \geq 1$.

Proof: By Lemma 1 and assumed BFS algorithm, we can use $O(\log^\alpha n)$ time and n^β processors to find a BFS tree on a face incidence graph. Then Miller's algorithm finds a $\sqrt{dn}$-cycle separator in $O(\log n)$ time using n processors. So, a $\sqrt{dn}$-cycle separator algorithm for undirected planar graphs needs at most $O(max(\log n, \log^\alpha n))$ time and $max(n, n^\beta)$ processors. ∎

Using standard Divide-and-Conquer techniques, we can find a family of $O(\sqrt{n})$ size separators on a planar graph in $O(\log n)$ iterations by applying a $O(\sqrt{n})$ size separator algorithm on the separated connected components of the graph concurrently in each iteration. This implementation immediately produces a near optimal parallel all-pairs shortest paths algorithm for planar graphs.

Theorem 2 *There is a parallel algorithm that finds all-pairs shortest paths on a planar graph in $O(\log^4 n)$ time using $n^2/\log^2 n$ processors.*

Proof: Since there is a parallel BFS algorithm that needs $O(\log^3 n)$ time and $n^{1.839}$ processors for a planar graph [13,17], by Theorem 1, a family of separators of size $O(\sqrt{n})$ on a planar graph can be found in $O(\log^4 n)$ time with $n^{1.839}$ processors. Then Pan and Reif's parallel nested dissection algorithm [13] can find all-pairs shortest paths in $O(\log^3 n)$ time using $n^2/\log^2 n$ processors, where we assume $\log^2 n \leq n^{0.161}$. ∎
Since the best sequential all-pairs shortest paths algorithm uses $O(n^2)$ time for planar graphs [4], the processor-time product of the above linear-processor DNC parallel all-pairs shortest paths algorithm is optimal within a $\log^4 n$ factor.

3 Ameliorating Sequences

Since a BFS algorithm for planar graphs and a Sep algorithm for planar graphs are reduced to each other, we want to use the two algorithms improving each other alternatively. To boot such an ameliorating sequence, we use a Sep algorithm that is flexible and fit a wide range number of processors. Our observation on BFS on face incidence graphs and Wan's Sep algorithm [17] are combined into the following algorithm as part of the base of our ameliorating sequence.

Theorem 3 [17] *Given a parallel algorithm performing BFS on a planar graph in $O(\log^\alpha n)$ time using n^β processors, there is a parallel algorithm that finds a vertex separator on a planar graph in $O(max(\log^2 n, \log^\alpha n))$ time using n^γ processors, where $1 \leq \gamma \leq \beta$. The size of the vertex separator is $O(n^{(2\beta-\gamma)/2\beta})$.*

Given a family of $s(n)$ size separators of a graph, G, Pan and Reif's algorithm finds the single-source shortest paths in G in $O(\log^3 n)$ time using $s^3(n)/\log^2 n$ processors. By Theorem 3, a parallel BFS algorithm using more than n processors implies a parallel Sep algorithm which uses fewer processors but finds separators larger than $O(\sqrt{n})$. An interesting problem is whether separators found by the new parallel Sep algorithm are still small enough so that they can be used to improve parallel BFS algorithm in turn. Under certain circumstances, the answer is positive. The following theorem formalizes this idea and creates an interesting mutual ameliorating relation between a BFS algorithm for planar graphs and a Sep algorithm for planar graphs.

Theorem 4 *Given a parallel algorithm finding a BFS tree in a planar graph in $O(\log^\alpha n)$ time using n^β processors, where $\alpha \geq 2$ and $\beta > 1.5$, there is a parallel algorithm that finds a small separator in an undirected planar in $O(\log^\alpha n)$ time using $n^{6\beta/(2\beta+3)}$ processors. The size of the separator is $n^{2\beta/(2\beta+3)}$. There is also a new parallel BFS algorithm for planar graphs using $O(\log^{\alpha+1} n)$ time and $n^{6\beta/(2\beta+3)}$ processors.*

Proof: The key point is deciding how many processors should be used for a new $s(n)$-separator algorithm so that it improves a parallel BFS algorithm as much as possible.

Because the expected new parallel BFS algorithm will use $O(s^3(n)/\log^2 n)$ processors, the new parallel $s(n)$-separator algorithm should make full use of that many processors too. If we let n^x denote the processor number we are looking for, by Theorem 3, our observation can be formalized as two equations: $s^3(n)=n^x$ and $s(n)=n^{(2\beta-x)/2\beta}$, where, for the sake of convenience, we only assume a weaker reduction using $O(\log^3 n)$ time and $s^3(n)$ processors to reduce the BFS problem to the Sep problem. The solution of the equations is $s(n)=n^{2\beta/(2\beta+3)}$. Based on this analysis, first, a family of separators of size $s(n)$ on a planar graph can be found in $O(\log^{\alpha+1} n)$ time with $O(s^3(n))=O(n^{6\beta/(2\beta+3)})$ processors. Then with same number of processors, Pan and Reif's algorithm can be used to find a BFS tree or calculate single-source shortest paths in the graph in $O(\log^3 n)$ time. ■

We can successively use the above ameliorating technique to get a sequence of BFS algorithms and Sep algorithms to approximate $O(n^{1.5})$ processor DNC BFS algorithm and DNC $\sqrt{dn}$-cycle separator algorithm.

Corollary 1 *Given a positive constant K, there is a sequence of K pairs of BFS algorithms and $O(\sqrt{n})$ size Sep algorithms for planar graphs. The ith BFS algorithms uses $O(\log^{i+2} n)$ time and n^{β_i} processors and the ith Sep algorithms uses $O(\log^{i+1} n)$ time and n^{β_i} processors. The index numbers are defined as $\beta_{i+1}=\beta_i/(\beta_i+3)$ for $\beta_0=2.376$.*

Proof: Use Miller's algorithm [11] as the first Sep algorithm and apply Theorem 4 for K times. ■

If i is not bound by a constant, a multiplicative factor of c^i should be added into the time complexities of the above BFS algorithms and Sep algorithms for a constant c. If we let K grow, $O(n^{1.5})$ processor complexity can be achieved by $K=O(\log\log n)$, but the BFS algorithms and Sep algorithms will not be poly-log time algorithms any more. Let $\epsilon_i=1.5-\beta_i$, since $\epsilon_{i+1}<\epsilon_i/2$, the processor number is converged to $n^{1.5}$ in a reasonable speed, e.g., $\beta_1=1.839$, $\beta_2=1.652$, $\beta_3=1.572$, $\beta_4=1.535$, $\beta_5=1.517$, $\beta_6=1.509$.

Corollary 2 *There is a finite sequence of parallel algorithms that find a cycle separator on an undirected planar graph in $O(\log^{i+2} n)$ time using n processors. The size of the cycle separator is $O(max(\sqrt{dn}, n^{(2\beta_i-1)/2\beta_i}))$.*

Proof: Use the new BFS algorithm in Wan's algorithm [17]. ■

When β_i approaches toward 1.5, the break point of separator sizes in the above corollary is close to $n^{1/3}$.

Since BFS or single-source shortest paths algorithm is a bottle neck in many parallel algorithms, these algorithms are improved by our algorithms accordingly, for example, now a maximum flow in a planar graph can be found in $O(\log^{i+2} n)$ time with n^{β_i} processors for any constant i [12].

Corollary 3 *If there is a DNC algorithm that can perform a BFS using $O(\log^{\alpha} n)$ time and $s^{\gamma}(n)$ processors when a family of $s(n)$ size separators is given, there is a finite sequence of BFS algorithm and Sep algorithm pairs. The ith BFS algorithm uses $O(\log^{i+\alpha} n)$ time and n^{β_i} processors and the ith Sep algorithm uses $O(\log^{i+\alpha-1} n)$ time and n^{β_i} processors. The index numbers are defined as $\beta_{i+1}=2\gamma\beta_i/(2\beta_i+\gamma)$ for $\beta_0=\gamma$.*

Proof: The proof is similar to the proof of Theorem 4. The only difference is that we let $\gamma=3$ previously. ■

When i grows, the processor complexities, n^{β_i}, converge to $n^{\gamma/2}$.

Remarks Through our ameliorating sequence, we have reduced the BFS problem and the small cycle separator problem to each other so that an improved algorithm for one of them may imply an improved algorithm for the other accordingly. It is not clear whether $s^2(n)$ processors are enough for a DNC BFS algorithm designed for planar graphs so that a near linear-processor DNC BFS algorithm can be achieved directly by the diagram described in the paper. How to find linear-processor BFS parallel algorithms for general graphs or planar graphs, with or without our approach, is still among the most interesting open problems in parallel graph algorithms.

Acknowledgement. I am grateful to Gregory Shannon for his valuable comments, reading the drafts of this paper, and encouragements.

References

[1] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, pages 265–273, October 1983.

[3] H. Djidjev. On the problem of partitioning planar graphs. *SIAM Journal on Algebraic and Discrete Methods*, 3(2):229–240, June 1982.

[4] G. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, December 1987.

[5] H. Gazit. *Processor Efficient Parallel Graph Algorithms.* PhD thesis, University of Southern California, Los Angeles, CA, August 1988.

[6] H. Gazit and G. Miller. A parallel algorithm for finding a separator in planar graphs. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 238–251, 1987.

[7] H. Gazit and G. L. Miller. An improved parallel algorithm that computes the bfs numbering of a directed graph. *Information Processing Letters*, 28(2):61–65, June 1988.

[8] R. Karp and V. Ramachandran. *A Survey of Parallel Algorithms for Shared-Memory Machines.* Technical Report UCB/CSD 88/408, Computer Science Division (EECS), University of California, Berkeley, CA, March 1988. To appear in the *Handbook of Theoretical Computer Science* by North-Holland.

[9] R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, August 1980.

[10] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Algebraic and Discrete Methods*, 36(2):177–189, April 1979.

[11] G. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32(3):265–279, June 1986.

[12] G. Miller and J. Naor. Flow in planar graphs with mutiple sources and sinks. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 112–117, 1989.

[13] V. Pan and J. Reif. Fast and efficient solution of path algorithm problems. *Journal of Computer and System Sciences*, 38(3):494–591, June 1989.

[14] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentic-Hall, 1982.

[15] V. Ramachandran and J. Reif. An optimal parallel algorithm for graph planarity. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 282–287, 1989.

[16] R. Tarjan. *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, 1983.

[17] F. Wan. *A Linear-Processor Algorithm for Finding Small Cycle Separators on Undirected Planar Graphs.* Technical Report TR-308, Department of Computer Science, Indiana University, Bloomington, IN, May 1990.

GRAPH LAYOUT USING QUEUES

Lenwood S. Heath
Department of Computer Science
Virginia Polytechnic Institute
Blacksburg, VA 24061

Frank Thomson Leighton
Department of Mathematics
Massachusetts Institute of Technology
Cambridge, MA 02139

Arnold L. Rosenberg
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Abstract

We study the problem of laying out the edges of a graph using queues. We describe a number of basic results about queue layouts of graphs, and we contrast these results with their analogues for stack layouts of graphs (the book-embedding problem).

1. Introduction

We study the use of queues to compute linear layouts of graphs. Our study and the particular questions we focus on have a tripartite motivation.

Comparing Stacks and Queues. A recurring theme in the study of computing mechanisms compares and contrasts the relative powers of stacks and queues. In computability theory, it is well known that two stacks endow an off-line finite-state control with the full power of a Turing machine, whereas a single stack does not; in contrast, a single queue suffice to yield the full power of a Turing machine. In quite another domain, Tarjan [17] characterizes the numbers of stacks and queues needed to compute given permutations; cf. Section 3. In a similar vein, Even and Itai [5] use stacks and queues to linearize graphs in much the way that we study here. Indeed, a major focus of our study will contrast queue-based layouts of graphs with stack-based layouts. Stack–based layouts have been studied extensively, under the aegis of the problem of *embedding graphs in books* [1,4].

The DIOGENES Design Methodology. In DIOGENES [15], an array of communicating processors is implemented in a conceptual line and some number of hardware queues and/or stacks pass over the entire line. The queues and/or stacks implement the communication links among processors in such a way that faulty processors are ignored, and all good processors are utilized. If the processors and their connections are represented by an undirected graph, then the DIOGENES layout problem is equivalent to a graph layout problem, where edges are assigned to conceptual queues and/or stacks. The variant of DIOGENES in which only stacks are used is one motivation for the studies of the book embedding problem: Bernhart and Kainen [1], Buss and Shor [2], Chung, Leighton and Rosenberg [4], Heath [8,9], and Yannakakis [18]. Note that only [15] has considered queues before. This research intends to investigate the same issues for queues as [4] does for stacks. In particular, we find significant points of departure between queue and stack layouts.

Scheduling Parallel Processors. Consider the following simple model of scheduling parallel computations in an architecture-independent fashion; cf. [14]. One represents the computation to be scheduled as a dag whose nodes represent the processes to be executed and whose arcs indicate computational dependencies: a process-node cannot be executed until all of its predecessors in the dag have been. Processes are queued up in a FIFO *Processor Queue* (PQ) as they become eligible for execution; each idle processor "grabs" the process at the head of the PQ. Our study focuses on the management of data in this scenario: where will the inputs to process P be when P is "grabbed" by a processor? Our solution is to have the PQ be coordinated with a *Data Manager* (DM) which itself is a collection of FIFO queues: When a process terminates, it places its "outputs" on the queues of the DM in such a way that when process P is "grabbed" by a processor, all inputs to P are at the heads of the DM queues. Our queue-based graph linearization problem idealizes this approach to the scheduling problem: The computation dag is the graph to be linearized; the linearization process implicitly specifies the loading of the PQ; the queues that control the linearization comprise the DM. In this abstract we idealize the problem even a step further by replacing the computation dag by an ordinary graph. In subsequent work, we plan to study a more faithful version of the scheduling problem.

Our study of queue layouts has yielded a number of surprises. We summarize the highlights that are included in this abstract. A new class of planar graphs, arched leveled-planar graphs, is shown to be a characterization of the 1-queue graphs. While 1-stack (outerplanar) graphs are easy to recognize, the recognition problem for 1-queue graphs is NP-complete. Any 1-queue graph can be laid out with 2 stacks, and any 1-stack graph can be laid out with 2 queues. An obvious generalization of these results fails: there is a class of graphs, the ternary n-cubes, that requires exponentially more stacks than queues. We show an apparent tradeoff between the queuenumber and the queuewidth for queue layouts of complete binary trees. In a table at the end, we summarize our queuenumber results for some special graphs.

2. Definitions

A *k-queue layout* QL of an undirected graph $G = (V, E)$ consists of a linear order of V and an assignment of each edge in E to exactly one of k queues, $q_1, \ldots, q_k$. We use $\sigma = 1, 2, \ldots, n$ to denote the order. Each queue q_j operates as follows. The vertices of V are scanned in left-to-right (ascending) order. When i is encountered, any edges assigned to q_j that have i as their right endpoint must be at the front of the queue and are removed (dequeued). Any edges assigned to q_j that have i as left vertex are placed on the back of the queue (enqueued) in ascending order of their right vertices. k is the *queuenumber* of the layout. The *queuenumber* of G, $\mathrm{QN}(G)$, is the smallest k such that G has a k-queue layout; G is said to be a *k-queue* graph. Let $w(i, q_j)$ be the number of edges in q_j just before i is encountered. Then the *queuewidth* of q_j is $\mathrm{QW}(q_j) = \max_{i \in V} w(i, q_j)$. The *maximum queuewidth* of the layout is $\mathrm{QW}(QL) = \max_j \mathrm{QW}(q_j)$. The *cumulative queuewidth* of the layout is $\mathrm{CQW}(QL) = \sum_j \mathrm{QW}(q_j)$.

A *k-stack layout SL* of an undirected graph consists of a linear order of V and an assignment of each edge in E to exactly one of k stacks, $s_1, \ldots, s_k$. Each stack s_j operates as follows. The vertices of V are scanned in left-to-right (ascending) order. When i is encountered, any edges assigned to s_j that have i as their right endpoint must be on the top of the stack and are removed (popped). Any edges assigned to s_j that have i as left vertex are placed on the top of the stack (pushed) in descending order of their right vertices. k is the *stacknumber* of the layout. The *stacknumber* of G, $\mathrm{SN}(G)$, is the smallest k such that G has a k-stack layout.

3. Fixed-Order Layouts

In this section, we fix an order $\sigma = 1, 2, \ldots, n$ of V and examine the difficulty of minimizing the number of queue or stacks required to complete σ to a layout. We concentrate on sets of edges that are obstacles to minimizing the number of stacks or queues.

A *k-rainbow* is a set of k edges $\{e_i = (r_i, s_i), 1 \le i \le k\}$ such that

$$r_1 < r_2 < \cdots < r_{k-1} < r_k < s_k < s_{k-1} < \cdots < s_2 < s_1;$$

in other words, a rainbow is a *nested* matching.

A *k-twist* is a set of k edges $\{e_i = (r_i, s_i), 1 \le i \le k\}$ such that

$$r_1 < r_2 < \cdots < r_{k-1} < r_k < s_1 < s_2 < \cdots < s_{k-1} < s_k;$$

in other words, a twist is a fully intersecting matching.

If σ has a k-rainbow (k-twist), then any queue (stack) layout of σ requires at least k queues (stacks). In fact, the size of the largest rainbow in σ exactly determines the queuenumber of σ:

Theorem 3.1 *If σ has no rainbow of more than k edges, then there is a k-queue layout for σ. Such a layout can be found in time $O(|E| \log \log n)$.*

For the fixed order σ, assigning edges to stacks is equivalent to coloring circle graphs ([4]). While it is possible to determine the largest twist size for σ in polynomial time (Hsu [10]), minimizing the number of stacks cannot be done in polynomial time unless $P = NP$, because coloring circle graphs is NP-complete (Garey, Johnson, Miller and Papadimitriou [7]). Therefore, optimizing fixed-order queue layouts is easier than optimizing fixed-order stack layouts.

4. Characterizing 1-queue Graphs

The characterization of 1-stack graphs was given in [1].

Proposition 4.1 [1] *G is a 1-stack graph if and only if G is outerplanar.*

(An *outerplanar* graph is a planar graph having a planar embedding in which all vertices appear on a common face.) We show that the 1-queue graphs are also planar graphs that have a particular kind of planar embedding.

Consider the normal cartesian (x, y) coordinate system for the plane. For i an integer, let ℓ_i be the vertical line defined by $\ell_i = \{(i, y) \mid y \in \mathbf{R}\}$. A graph $G = (V, E)$ is a *leveled-planar graph* if V can be partitioned into levels $V_1, V_2, \ldots, V_m$ and G can be embedded in the plane such that all vertices of V_i are on the line ℓ_i, each edge in E is embedded as a straight line segment wholly between ℓ_i and ℓ_{i+1} for some i, and the embedding is a valid planar embedding for G (i.e., no edges cross). Note that the leveled-planar embedding of a leveled-planar graph is not necessarily unique.

A leveled-planar embedding induces an order (the *induced order*) on V as follows. As i takes the values $1, 2, \ldots, m$, scan line ℓ_i from bottom to top. Label the vertices $1, 2, \ldots, n$ as they are encountered. For $1 \leq i \leq m$, let b_i be the (bottom) first vertex in level i, and let t_i be the (top) last. Let s_i be the first vertex in level i that is adjacent to some vertex in level $i+1$, or, if there are no edges between levels i and $i+1$, let $s_i = t_i$. Consider augmenting G with new edges. A *level i arch* for G is an edge connecting t_i with j, where $b_i \leq j \leq \min(t_i - 1, s_i)$. A leveled-planar graph G, augmented by any number of arches, can be embedded in the plane by drawing the arches around level 1; because of the leveling, the arches do not cross. A leveled-planar graph augmented by (zero or more) arches is called an *arched leveled-planar graph.* The edges that are not arches are called *leveled edges.* An arched leveled-planar graph that cannot be augmented with further arches or leveled edges is *maximal.* See Figure 1 for an example. The above definitions for b_i, s_i, and t_i will be used throughout the abstract to refer to vertices in arched leveled-planar graphs.

Theorem 4.1 (Characterization Theorem) *A graph G is a 1-queue graph if and only if G is an arched leveled-planar graph.*

Proof: First suppose G is an arched leveled-planar graph. Then the induced order of any leveled-planar embedding of G yields a 1-queue layout of G.

Now suppose $G = (V, E)$ is a 1-queue graph. Let $\sigma = 1, 2, \ldots, n$ be the order of a 1-queue layout of G. It suffices to describe an arched leveled-planar embedding of G. Without loss of generality, we may assume that G is connected.

Partition V as follows. $V_1 = \{1\}$, and $b_1 = s_1 = t_1 = 1$. For $i > 1$, until each vertex is placed in some set, let t_i be the rightmost vertex incident to some vertex in V_{i-1}. Let $b_i = t_{i-1} + 1$. Let $V_i = \{b_i, \ldots, t_i\}$. Let s_i be the leftmost vertex in V_i that is adjacent to some vertex to the right of t_i; let $s_i = t_i$ if $t_i = n$.

Let the resulting partition be $V_1, V_2, \ldots, V_m$. This partition consists of breaking the sequence σ into m contiguous subsequences that end at $1 = t_1, t_2, \ldots, t_m = n$, respectively. By the construction of V_i, it is clear that there is no edge from a vertex in V_i to

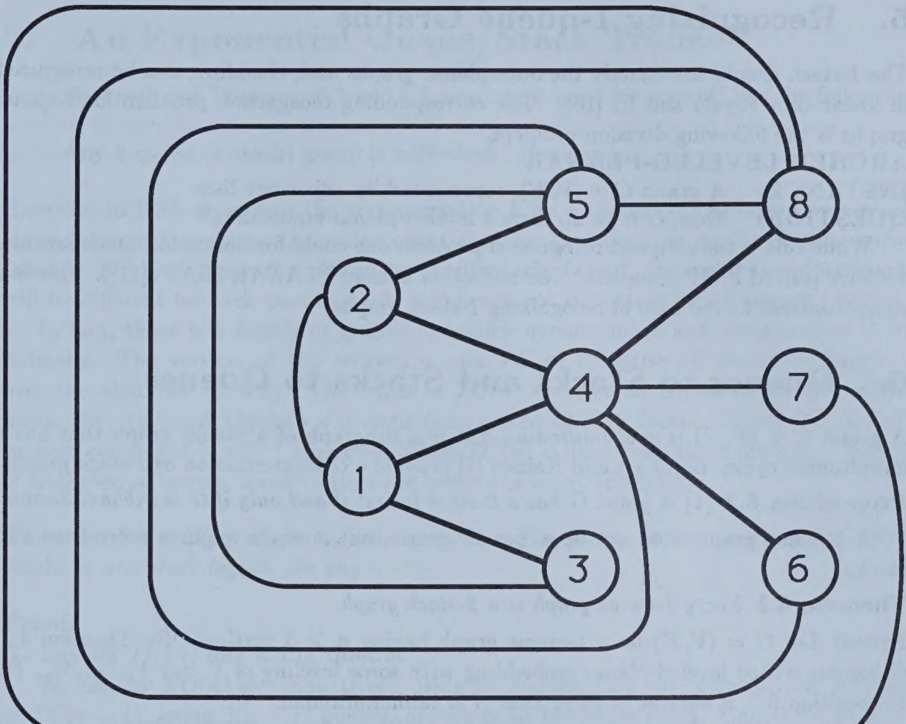


Figure 1: A maximal arched leveled-planar graph.

a vertex in V_j if $|i-j| \geq 2$. Let E_ℓ be the subset of E consisting of edges that connect vertices at consecutive levels; that is,

$$E_\ell = E \cap \bigcup_{1 \leq i < m} V_i \times V_{i+1}.$$

First construct a leveled-planar embedding of $G_\ell = (V, E_\ell)$. Place the vertices of V_i on line ℓ_i in the order $b_i, b_i + 1, \ldots, t_i$ from bottom to top. Draw the edges in E_ℓ as line segments. Two of these line segments might cross only if they end on two adjacent lines ℓ_i and ℓ_{i+1}. Let (p_1, q_1), $p_1 < q_1$ and (p_2, q_2), $p_1 < q_2$ be the edges corresponding to two line segments that cross. Then $p_1 < p_2 < q_2 < q_1$, that is, the edges nest. That is a contradiction to σ being the order of a 1-queue embedding of G.

It remains to show that $E - E_\ell$ contains only arches for G_ℓ. Let $(p_3, q_3) \in E - E_\ell$, where $p_3, q_3 \in V_i$, $p_3 < q_3$. Clearly, $p_3 \leq \min(t_i - 1, s_i)$ since otherwise there is an edge from s_i to some vertex in V_{i+1} that nests over (p_3, q_3). Since t_i is adjacent to some vertex $x \in V_{i-1}$, we must have $q_3 = t_i$, for otherwise (x, t_i) and (p_3, q_3) nest. We conclude that $E - E_\ell$ contains only arches and that we have constructed an arched leveled-planar embedding of G. □

5. Recognizing 1-queue Graphs

The 1-stack graphs are exactly the outerplanar graphs and, therefore, can be recognized in linear time (Sysło and Iri [16]). The corresponding recognition problem for 1-queue graphs is the following decision problem.

ARCHED LEVELED-PLANAR

INSTANCE: A graph $G = (V, E)$, represented by adjacency lists.

QUESTION: Does G have an arched leveled-planar embedding?

While this is the simplest recognition problem one could formulate for queue layouts, we have proved it NP-complete. The reduction is from PLANAR 3SAT ([11]). This is a sharp contrast to the ease of recognizing 1-stack graphs.

6. Queues to Stacks and Stacks to Queues

A graph $G = (V, E)$ is *subhamiltonian* if it is a subgraph of a planar graph that has a hamiltonian cycle. Bernhart and Kainen [1] provide a characterization of 2-stack graphs.

Proposition 6.2 **[1]** *A graph G has a 2-stack layout if and only if G is subhamiltonian.*

A 1-queue graph need not be a 1-stack graph, but it never requires more than two stacks.

Theorem 6.2 *Every 1-queue graph is a 2-stack graph.*

Proof: Let $G = (V, E)$ be a 1-queue graph having $n \geq 3$ vertices. By Theorem 4.1, G has an arched leveled-planar embedding with some leveling of V, say $V_1, \ldots, V_m$. By Proposition 6.2, it suffices to show that G is subhamiltonian.

Because the stacknumber of a graph equals the maximum stacknumber of any of its biconnected components [4], we may assume that G is biconnected, so, in particular, none of the levels $V_2, \ldots, V_{m-1}$ is a singleton. We may also assume that G is a maximal arched leveled-planar graph. For each level i, add the *vertical* edges $(p, p+1)$, $b_i \leq p \leq t_i - 1$, that is, the edges that go along the line ℓ_i, connecting consecutive vertices of V_i. Let the resulting graph be $G' = (V, E')$. Clearly G' is planar; we claim that it is hamiltonian.

Note that when $|V_i| > 2$, the vertical edges on level i together with the arch (b_i, t_i) form a cycle on V_i. Call these edges the *level-i cycle edges.* These cycles on levels are nested in the planar embedding. Our strategy is to connect each pair of consecutive cycles by two leveled edges to obtain a hamiltonian cycle for G'.

By an induction on $m - i \geq 1$, we show that there is a particular kind of spanning cycle for levels $V_i, V_{i+1}, \ldots, V_m$. The inductive hypothesis is that there is a cycle C spanning levels $V_i, \ldots, V_m$ such that all but one of the level-i cycle edges are in C; if $|V_i| = 2$, then C contains the edge (b_i, t_i) (which is considered to be both a vertical edge and an arch). We omit the case analysis necessary to complete the inductive proof that G' is hamiltonian. By Proposition 6.2, the theorem follows. □

Similarly, a 1-stack graph need not be a 1-queue graph, but it never requires more than two queues.

Theorem 6.3 *Any 1-stack graph is a 2-queue graph.*

7. An Exponential Queue/Stack Tradeoff

From the results of Theorems 6.2 and 6.3, one might hope for a result like the following:

> Any k-queue (k-stack) graph is a $2k$-stack ($2k$-queue).

However, in both theorems, the transformation from a queue (stack) layout to a stack (queue) layout transforms the vertex order in a way that depends on the original queue (stack). But for a general multi-queue (multi-stack) layout, the order transformations will be different for each queue (stack) and hence not consistent for all queues (stacks).

In fact, there is a family of graphs for which queuenumber and stacknumber differ radically. The vertices of the *ternary n-cube* $TC(n)$ comprise all strings of length n over the alphabet $\{0,1,2\}$. The edges of $TC(n)$ connect all triples of vertices of the forms $x0y$, $x1y$, and $x2y$ into a triangle (i.e., a copy of K_3). Hence, $TC(n)$ has $N = 3^n$ vertices connected in $n3^{n-1}$ triangles. When $TC(n)$ is linearized, there are $n3^{n-1}$ *centers* of triangles; of course, some vertices are centers of many triangles.

Theorem 7.4 *$TC(n)$ admits a queue layout using $2n$ queues but requires at least $\Omega(N^{\frac{\alpha}{3}})$ stacks in any stack layout, for any $\alpha < \frac{1}{3}$.*

Proof:

A. Laying $TC(n)$ out using queues

We can lay $TC(n)$ out inductively, using $2n$ queues. Lay out consecutively 3 copies of $TC(n-1)$, using $2(n-1)$ queues, all copies in the same order. Call them copy A, copy B, and copy C, from left to right. Now assign one new queue for the A-C edges and one new queue for the A-B edges and the B-C edges. This clearly works.

B. Laying $TC(n)$ out using stacks

An Upper Bound. A stack layout of $TC(n)$ using $O((N \log N)^{1/2})$ stacks follows immediately from Theorem 2.2 in [12]. Since the proof of that theorem is nonconstructive, it does not provide an explicit stack layout of $TC(n)$.

A Lower Bound. We claim that any way of linearizing $TC(n)$ must use $\Omega(N^{\frac{\alpha}{3}})$ stacks, for any $\alpha < \frac{1}{3}$. Here are the bones of the argument, which reduces the sum-of-triangles layout from [4] to the current problem. Recall that when $TC(n)$ is linearized, there are $nN/3$ *centers* of triangles among the N vertices (if a triangle has vertices u, v, w, and they appear in that order in the linearization, then v is the center). Focus on an integer $K = N^{\alpha}$, and partition the given arbitrary layout of $TC(n)$ into disjoint windows of size K. Some one window W must contain at least $nK/3$ center vertices (note that, in this count, we allow a single vertex to be the center of many triangles). We want to bound (from above) the number of triangles of $TC(n)$ that have a center in window W AND ALSO have another vertex in W. When this happens, say that the "other" vertex (i.e., the non-center) *spoils* the triangle. We want to determine the largest possible number of edges that could connect vertices within W (i.e., that do not exit W). We show that at most αnK edges remain within W (this is the ternary hypercube version of a result for the binary hypercube of Chung, Füredi, Graham and Seymour [3]). Each such edge can spoil at most 1 triangle. Hence, W contains at least $(1/3 - \alpha)nK$ centers of unspoiled triangles. Any vertex can be the center of at most n triangles. Hence, W contains at least $(1/3 - \alpha)K$ distinct vertices that are centers of unspoiled triangles. Fix $\alpha < 1/3$. By the sum-of-triangles lower bound [4], the result follows. □

The exponential magnitude of this tradeoff is certainly a surprise. Compare this result to the fact that the queuenumber and stacknumber [4] of the boolean n-cube are both $\Theta(n)$. A substantial tradeoff in the opposite direction has eluded us.

Often, good queue layouts seem easier to obtain than good stack layouts. Planar graphs may be an exception to this. In harmony with the fact that planar graphs can be laid out in a bounded number of stacks (Yannakakis [18]), we conjecture

Conjecture. Planar graphs can be laid out in a bounded number of queues.

8. Queue/Queuewidth Tradeoffs for Complete Binary Trees

In this section, we provide evidence of an apparent tradeoff between queuenumber and queuewidth for queue layouts of complete binary trees. The first theorem suggests a tradeoff between diameter and queuewidth for 1-queue graphs.

Theorem 8.1 *Suppose G is a connected 1-queue graph having diameter D. Let QL be a 1-queue layout of G. Then,*

$$\mathrm{QW}(QL) \geq |E|/(2D+1).$$

Corollary 8.1 *Any 1-queue layout of a depth-d complete binary tree with $n = 2^d$ leaves has queuewidth at least $(2n-2)/(4d+1) = \Omega(n/\log n)$.*

The breadth-first layout of T starting at the root is a 1-queue layout of queuewidth n. The lower bound on queuewidth in the corollary is close to this upper bound. We do not know how to achieve this lower bound and doing so appears difficult. These bounds are much larger than the $O(\log n)$ stackwidth of a 1-stack layout of T ([4]). The higher width of queue layouts over stack layouts suggests that stack layouts of trees are preferable to queue layouts. The question arises whether a lower cumulative queuewidth can be achieved by using additional queues. The following theorem answers the question affirmatively.

Theorem 8.2 *A depth-d complete binary tree T with $n = 2^d$ leaves has a k-queue layout QL with $\mathrm{CQW}(QL) = O(kn^{1/k})$.*

Proof: We give the proof for $k = 2$. To simplify the construction, we assume that $d = 2d'$ is even. Let $n' = 2^{d'} = \sqrt{n}$. Let T_p be the upper $d'+1$ levels of T, and let $1, 2, \ldots, n'$ be the leaves of T_p in canonical order. Each i is the root of a subtree T_i of depth d'. Order the vertices of T_p in the obvious breadth-first order so that $1, \ldots, n'$ appear rightmost in the order. For each $i, 1 \leq i \leq n'$, place the vertices of T_i in breadth-first order immediately to the right of its root i. Assign the edges of T_p to one queue and the edges of $T_1, T_2, \ldots, T_{n'}$ to a second queue. Each queue has queuewidth $n' = \sqrt{n}$. The cumulative queuewidth of the 2-queue layout is $2n' = O(2n^{1/2})$.

The details of the proof for $k > 2$ are left to the reader. □

We have no lower bounds on cumulative queuewidth for multi-queue embeddings of T. Thus we do not know whether there is a real tradeoff between queuenumber and queuewidth here.

References

[1] F. Bernhart and B. Kainen, "The book thickness of a graph," *Journal of Combinatorial Theory B,* vol. 27, 1979, pp. 320-331.

[2] J. Buss and P. Shor, "On the pagenumber of planar graphs," *Proceedings of the 16th ACM Symposium on Theory of Computing,* pp. 98-100.

[3] F. R. K. Chung, Z. Füredi, R. L. Graham and P. Seymour, "On induced subgraphs of the cube," *Journal of Combinatorial Theory A,* vol. 49, 1988, pp. 180-187.

[4] F. R. K. Chung, F. T. Leighton and A. L. Rosenberg, "Embedding graphs in books: a layout problem with applications to VLSI design," *SIAM Journal on Algebraic and Discrete Methods,* vol. 8, 1987, pp. 33-58.

[5] S. Even and A. Itai, "Queues, stacks and graphs," In *Theory of Machines and Computations* (Z. Kohavi and A. Paz eds.) Academic Press, NY, 1971, pp. 71-86.

[6] R. Games, "Optimal book embeddings of the FFT, Benes, and barrel shifter networks," *Algorithmica,* Vol. 1, 1986, pp. 233-250.

[7] M. R. Garey, D. S. Johnson, G. L. Miller and C. H. Papadimitriou, "The complexity of coloring circular arcs and chords," *SIAM Journal on Algebraic and Discrete Methods,* Vol. 1, 1980, pp. 216-227.

[8] L. S. Heath, "Embedding planar graphs in seven pages," *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science,* 1984, pp. 74-83.

[9] L. S. Heath, "Algorithms for embedding graphs in books," Ph.D. dissertation, University of North Carolina at Chapel Hill, 1985.

[10] W.-L. Hsu, "Maximum weight clique algorithms for circular-arc graphs and circle graphs," *SIAM Journal on Computing,* Vol. 14, 1985, pp. 224-231.

[11] D. Lichtenstein, "Planar formulae and their uses," *SIAM Journal on Computing,* Vol. 11, 1982, pp. 329-343.

[12] S. M. Malitz, *Graphs with E edges have pagenumber* $O(\sqrt{E})$, submitted for publication, 1989.

[13] D. J. Muder, M. L. Weaver, and D. B. West, "Pagenumber of complete bipartite graphs," *Journal of Graph Theory,* Vol. 12, 1988, pp. 469-489.

[14] C. H. Papadimitriou and M. Yannakakis, "Towards an architecture-independent analysis of parallel algorithms," *Proceedings of the 20th Annual ACM Symposium on Theory of Computing,* 1988, pp. 510-513.

[15] A. L. Rosenberg, "The DIOGENES approach to testable fault-tolerant arrays of processors," *IEEE Transactions on Computers,* vol. C-32, 1983, pp. 902-910.

[16] M. M. Sysło and M. Iri, "Efficient outerplanarity testing," *Fundamenta Informaticae,* Vol. 2, 1979, pp. 261-275.

[17] R. E. Tarjan, "Sorting using networks of queues and stacks," *Journal of the ACM,* vol. 19, 1972, pp. 341-346.

[18] M. Yannakakis, "Four pages are necessary and sufficient for planar graphs," *Proceedings of the 18th Annual ACM Symposium on Theory of Computing,* 1986, pp. 104-108.

Graph Class	Queuenumber (Our Results)	Stacknumber
Trees	1	1 [4]
X-trees	2	2 [4]
DeBruijn Graph	2	Unknown
Complete Graph K_n	$\lfloor n/2 \rfloor$	$\lfloor n/2 \rfloor$ [4]
Complete Bipartite Graph $K_{m,n}$	$\min(\lceil m/2 \rceil, \lceil n/2 \rceil)$ (Exact)	$\leq \lceil (m+2n)/4 \rceil$ [13]
FFT Network	2	3 [6]
Benes Network	2	3 [6]
Boolean n-cube	$\leq n-1$	$\leq n-1$ [4]
Planar Graphs	Unknown (Conjecture bounded)	4 [18]

Table 1: Queuenumbers of Specific Graphs

An Efficient Algorithm for the Maximum k-chain Problem [1]

R. D. Lou and *M. Sarrafzadeh*
Department of Electrical Engineering and Computer Science
The Technological Institute
Northwestern University
Evanston, IL 60208

Abstract

Given a two-dimensional point set ρ, a *chain* C is a subset of ρ in which for every two points one is dominated by the other. A *k-chain* is a subset of ρ that can be partitioned into k chains. Size of a k-chain is the total number of its points. A k-chain with maximum size, among all possible k-chains, is called a *maximum k-chain.* First we study geometric properties of k-chains, then we present a $\Theta(n \log n)$ time algorithm for finding a maximum *three*-chain in a point set ρ, where $n = |\rho|$.

1 Introduction

Consider a point set $\rho = \{P_1, P_2, ..., P_n\}$ in the x-y plane, where $P_i = (x_i, y_i)$ with $x_i > 0$ and $y_i > 0$; by convention, $x_i > x_j$ for $i > j$ and all the y's are distinct. Following [PSh] we say P_i *dominates* P_j if $x_i > x_j$ and $y_i > y_j$. Points P_i and P_j are *crossing* if $x_i > x_j$ and $y_j > y_i$ or vice versa; otherwise, they are *noncrossing.* A subset of ρ is called a *chain* if its points are pairwise noncrossing. A subset is called a *k-chain* if it can be partitioned into k chains. *Size* of a k-chain is the number of its points. A *maximum chain* is a chain with maximum size among all possible chains. A *maximum k-chain* is a k-chain with maximum size among all possible k-chains. The problem of finding a maximum k-chain is equivalent to the following problems:

Graph-theoretic Problem : In a given representation of a permutation graph, as a family of chords, find a maximum k-partite subgraph. (The more general problem of finding a maximum k-partite subgraph in an overlap graph (i.e., a circle graph) is NP-hard [SLe1].)

Sequence Manipulation : Given a sequence of integers, find a maximum k-increasing subsequence. (A k-increasing subsequence is a subset that can be partitioned into k increasing subsequences.)

VLSI Layout : The maximum k-layer subset problem and the k-layer topological via minimization problem in a channel. Normally, k is a small constant (2, 3, or 4) [SLe1,SLe2,CL,RKN,SLo].

---

[1] This work was supported in part by the National Science Foundation under Grant MIP-8921540.

Unweighted maximum k-partite subgraph in transitive graphs (which includes the class of permutation graphs) was solved in $O(kn^2)$ time, where n is the number of vertices [Ga]. This result was extended to weighted case in [CL,SLo] (both algorithms run in $O(kn^2)$ time.) Employing the "geometric nature" of permutation graphs (i.e., solving the equivalent maximum k-chain problem), the following results have been obtained: A $\Theta(n \log n)$ time algorithm for finding a maximum one-chain has been proposed [AK]. This result was extended to find a maximum two-chain in $\Theta(n \log n)$ time [LSL]. In this paper, we study gemetric properties of k-chains. Then, we propose a $\Theta(n \log n)$ time algorithm for finding a maximum *three*-chain.

In the next section we present geometric properties of maximum chains. A brief outline of the algorithm is presented. For more details see [LS].

2 Geometric Properties of Chains

Consider a chain $C = \{P_{\pi_1}, P_{\pi_2}, ..., P_{\pi_m}\}$ in a point set ρ, where $x_{\pi_i} < x_{\pi_{i+1}}$, for $1 \leq i < n$. In the xy-plane, an *extended path* $\mathcal{P}(C)$ of C is a path vertically from $(x, y) = (x_{\pi_1}, -\infty)$ to P_{π_1}, then to (x_{π_2}, y_{π_1}), then to P_{π_2}, ..., P_{π_m}, and finally from P_{π_m} to (x_{π_m}, ∞). Extended path $\mathcal{P}(C)$ partitions the point set ρ into three subsets:

a) subset of points on the left-hand side of $\mathcal{P}(C)$;

b) chain C; and

c) subset of points on the right-hand side of $\mathcal{P}(C)$.

Given an h-chain H, we can partition H into h chains such that the i-th chain (from left to right) is on the left-hand side of the extended path of the $(i+1)$-st chain. We call such partition a *normal partition* of H, and each chain a *component chain* of H. (Clearly, normal partition of every h-chain can be found in linear time.) $H(i)$ represents the i-th, from left to right, component chain of H. In the following discussion, we express each h-chain H by $H = H(1) \cup H(2) \cup ... \cup H(h)$.

Let M_h represent a maximum h-chain and $M_h(i)$ represent the i-th component chain of M_h. The extended paths of the component chains of M_h (i.e., $\mathcal{P}(M_h(1))$, $\mathcal{P}(M_h(2))$, ... , and $\mathcal{P}(M_h(h))$) partition the point set ρ into $2h+1$ subsets:

a) h component chains of M_h;

b) $R_h(1)$, the subset of the points on the left-hand side of $\mathcal{P}(M_h(1))$;

c) $R_h(i)$, the subset of the points on the right-hand side of $\mathcal{P}(M_h(i-1))$ and on the left-hand side of $\mathcal{P}(M_h(i))$, for $2 \leq i \leq h$;

d) $R_h(h+1)$, the subset of the points on the right-hand side of $\mathcal{P}(M_h(h))$.

Consider a maximum h-chain M_h and an $(h+1)$-chain $F = F(1) \cup F(2) \cup \ldots \cup F(h+1)$ in point set ρ. We say F satisfies the *partition property* of M_h if:

a) $F(1)$ is a subset of $R_h(1) \cup M_h(1)$;

b) $F(i)$ is a subset of $M_h(i-1) \cup R_h(i) \cup M_h(i)$, for $2 \leq i \leq h$;

c) $F(h+1)$ is a subset of $M_h(h) \cup R_h(h+1)$.

Given a maximum h-chain M_h and a chain C, it is easy to construct an $(h+1)$-chain $F' = M_h \cup C$, such that F' satisfies the partition property of M_h. An example is shown in Figure 1.

Lemma 1 *Given a maximum h-chain M_h and an $(h+1)$-chain F in point set ρ, there exists a $(h+1)$-chain F' such that F' satisfies the partition property of M_h and $|F'| \geq |F|$.*

Proof: Let $F(j)$ be a component chain of F. $M_h \cap F(j)$ may not be empty. The total number of points in the other h component chains of F is less than or equal to that of M_h. Clearly, we can construct an $(h+1)$-chain F'=$M_h \cup F(j)$, such that F' satisfy the partition property of M_h and $|F'| \geq |F|$. □

The following lemma is a corollary of Lemma 1.

Lemma 2 *Given a maximum h-chain M_h in point set ρ, there exists at least one maximum $(h+1)$-chain M_{h+1} such that M_{h+1} satisfies the partition property of M_h.*

Consider a chain C in point set ρ. The highest point (i.e., with maximum y-coordinate) of C is called the *top point* of C. Similarly, the lowest point of C is called the *bottom point* of C. Assume $C = \{P_{\pi_1}, P_{\pi_2}, ..., P_{\pi_m}\}$. Let $C' = \{P_{\pi_i}, P_{\pi_{i+1}}, ..., P_{\pi_{i+j}}\}$ be a subset of C, such that there is no point in $C - C'$ that dominates some point in C' and is also dominated by some point in C'. Then we say C' is a *continuous portion* of C. Let C^1, C^2 and C^3 be three, not necessarily distinct, chains in ρ. Assume C^1 and C^2 have some common points. A point set D is called an *overlapped portion* of C^1 with C^2, if D has the following properties:

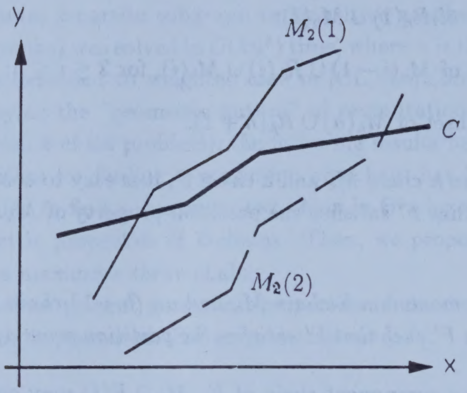
O1) D is a continuous portion of C^1 and C^2 (hence, D is also a subset of common points of C^1 and C^2);

O2) No other point set with property O1 contains D as a subset (i.e., D is maximal).

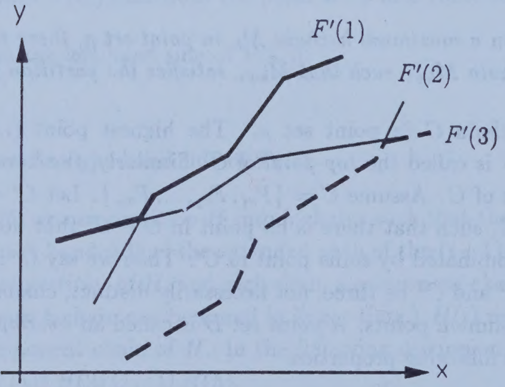
Similarly, we define overlapped portions of C^2 with C^3. Notice that, an overlapped portion of C^1 with C^2 may "overlap" with an overlapped portion of C^2 with C^3. With this definition, if M_{h+1} satisfies the partition property of M_h, then M_{h+1} has the following properties (to be proved below):

P1) All bottom points of M_h are bottom points of M_{h+1}.

P2) All top points of M_h are top points of M_{h+1}.



(a) before combination



(b) after combination

Figure 1: The combination of M_2 and C

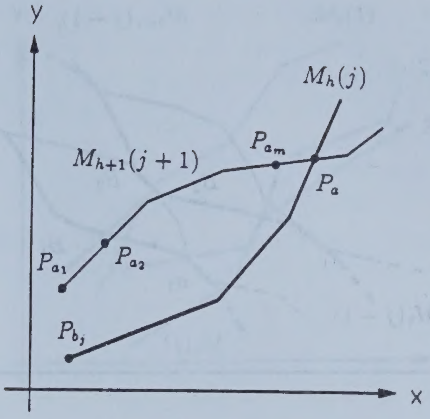


Figure 2: Proof of Lemma 3, P1 and P2

P3) On each component chain $M_h(j)$, its overlapped portions with $M_{h+1}(j)$ and with $M_{h+1}(j+1)$ appear alternatively from bottom to top.

Lemma 3 *If M_{h+1} satisfies the partition property of M_h, then M_{h+1} has properties P1, P2 and P3.*

Proof: 1)*For properties P1 and P2*:
Assume one bottom point P_{b_j} of component chains $M_h(j)$ is not a bottom point of any component chain of M_{h+1} (see Figure 2). Let the lowest point of $M_h(j) \cap M_{h+1}$ be P_a. Without loss of generality, assume P_a is on $M_{h+1}(j+1)$. Let the points on $M_{h+1}(j+1)$, lower than P_a, and not in M_h be P_{a_1}, P_{a_2}, ..., P_{a_m}, where $x_{a_1} < x_{a_2} < ... < x_{a_m}$. Let m' denote the number of the points on $M_h(j)$ below P_a. Because M_{h+1} is a maximum $(h+1)$-chain, $m \geq m'$. Because M_h is a maximum h-chain, $m \leq m'$. Thus, $m = m'$. Hence, we can delete points P_{a_1}, P_{a_2}, ..., P_{a_m} from M_{h+1} and add the points which are on $M_h(j)$ and below P_a to M_{h+1}. In the new M_{h+1}, P_{b_j} is the bottom point of $M_{h+1}(j)$. Repeating this procedure, we can make all bottom points of M_h bottom points of M_{h+1}. Similarly, we can make all top points of M_h top points of M_{h+1}.
2)*For property P3*:
Assume property P3 is not satisfied for maximum $(h+1)$-chain M_{h+1}. Let $M_h(j)$ be the leftmost component chain of M_h, such that property P3 is not satisfied (i.e., the overlapped portions of $M_h(j)$ with $M_{h+1}(j)$ and with $M_{h+1}(j+1)$ do not appear alternatively). Consider the following two cases:
Case i): Assume σ_1 and σ_2 are two overlapped portions of $M_h(j)$ with $M_{h+1}(j)$ and no overlapped portions of $M_h(j)$ with $M_{h+1}(j+1)$ is between σ_1 and σ_2 (see Figure 3). That is, there is no common point of $M_h(j)$ and $M_{h+1}(j+1)$ which dominates points in σ_1 and is dominated by points in σ_2.

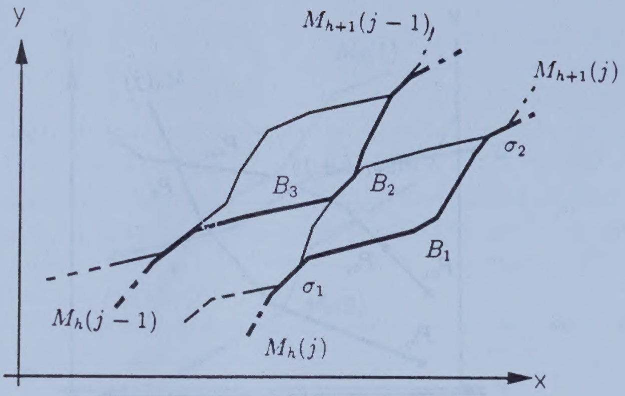


Figure 3: Proof of Lemma 3, P3

Let B_1 be the set of points in $M_h(j)$ and between σ_1 and σ_2. Let B_2 be the set of points in $M_{h+1}(j)$ between σ_1 and σ_2. According to properties P1 and P2, the top point and the bottom point of $M_h(j-1)$ are in M_{h+1}. Also according to the assumption that $M_h(j)$ is the leftmost component chain of M_h, P3 is false. Thus, in $M_h(j-1)$, let B_3 be the portion between two overlapped portions of $M_h(j-1)$ with $M_{h+1}(j-1)$.

If we delete all points in B_2 from $M_{h+1}(j)$ and add all points in B_1 to $M_{h+1}(j)$, the union of σ_1, B_1 and σ_2 becomes a new overlapped portion of $M_h(j)$ and $M_{h+1}(j)$. Because M_h is a maximum h-chain, $|B_1| \geq |B_2|$. Thus, the new M_{h+1} is still a maximum $(h+1)$-chain. However, B_3 is not in the new M_{h+1} and $M_h(j-1)$ is the leftmost component chain of M_h Thus property P3 is now satisfied. Repeateing the above procedure, we can make all component chains of M_h satisfy property P3.

Case ii): Assume σ_1 and σ_2 are two overlapped portions of $M_h(j)$ with $M_{h+1}(j+1)$ and no overlapped portions of $M_h(j)$ with $M_{h+1}(j)$ is between σ_1 and σ_2. Let B_1 be the set of points in $M_h(j)$ and between σ_1 and σ_2. Let B_2 be the set of points in $M_{h+1}(j+1)$ between σ_1 and σ_2. Let B_3 be the set of points in $M_{h+1}(j+1)$, between σ_1 and σ_2. As in Case i, we can make all component chains of M_h satisfies property P3. □

Given a maximum h-chain M_h in ρ, we call the points on M_h *marked points* and the other points in ρ *unmarked points.* A path, called *z-chain* of M_h (or, simply, z-chain), can be constructed as follows (see Figure 4 where thin lines correspond to the chains and thick lines correspond to the z-chain). The path has a starting point and an ending point. We trace points on the path from the starting point. According to the order that points are traced, we order them. Let P_i be a point on the path. Let P_j be the point next to P_i on the path. If P_j is an unmarked point then P_j must dominate P_i; otherwise, P_j must either dominate or be dominated by P_i. If P_j dominate P_i, then we say P_j is an *up-point*; otherwise, P_j is a *down-point.* The starting point is defined to

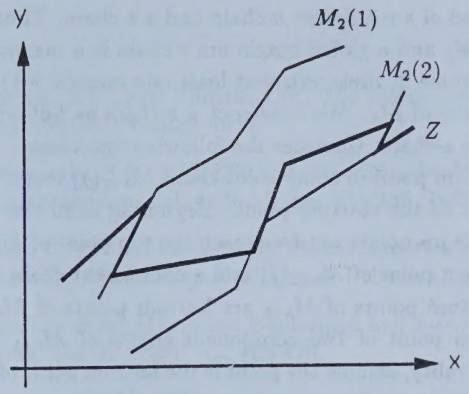


Figure 4: An example of z-chain

be an up-point. If P_i is an up-point and P_j is a down-point, we call P_i a *peak point.* If P_i is a down-point and P_j is an up-point, we call P_i a *valley point.* Each portion of points between a peak point and its following valley point in the path is called a *reverse chain.* We also define *z-size* of a z-chain as the number of unmarked up-points minus the number of marked non-valley down-points in the z-chain. A z-chain (of M_h) from point P_a to point P_b with maximum z-size among all z-chains with the same starting point P_a and ending point P_b, is called a *maximum z-chain* (of M_h) from P_a to P_b. A maximum z-chain with maximum z-size among all possible maximum z-chains, between all pairs of points in ρ, is called a *global maximum z-chain* (of M_h).

Given a maximum h-chain M_h and a z-chain Z in ρ, if we change unmarked up-points to be marked and non-valley down-points to be unmarked, the set of resultant marked points is called the *combination of M_h and Z.* Obviously, the size of the combination of M_h and Z is the sum of the size of M_h and the z-size of Z.

Lemma 4 *In a maximum z-chain, no point is repeated.*

Proof: Assume Z is a maximum z-chain and point P_a in Z appears twice. Then, there is a closed loop z-chain Z_c whose z-size is larger than 0. Clearly, the combination of M_h and Z_c is a h-chain with size larger than that of M_h. We reach a contradiction. □

Lemma 5 *The combination of a maximum h-chain M_h and a global maximum z-chain is a maximum $(h+1)$-chain.*

Proof: Clearly, the combination of a maximum h-chain and a z-chain is a $(h+1)$-chain. In the following, we will show that a maximum $(h+1)$-chain can always be obtained from the combination of a maximum h-chain and a z-chain. Thus, the combination of a maximum h-chain M_h and a global maximum z-chain is a maximum $(h+1)$-chain.

According to Lemma 2, there exists at least one max-$(h+1)$-chain M_{h+1} satisfying the partition property of M_h. We construct a z-chain as follows. First we choose the starting point of the z-chain. Consider the following two cases:

Case i): If the bottom point of component chain $M_{h+1}(j)$ is not a bottom point of M_h, we choose the point as the starting point. Beginning from the starting point, let the points on $M_{h+1}(j)$ be up-points until we reach the top point of $M_{h+1}(j)$ or reach a point P_a which is a common point of $M_{h+1}(j)$ and a component chain of M_h, say $M_h(j-1)$.

Case ii): If all bottom points of M_{h+1} are bottom points of M_h, we choose the point which is the bottom point of two component chains of M_{h+1} as the starting point. Without loss of generality, assume the point is the bottom point of $M_{h+1}(j)$, $M_{h+1}(j+1)$, and $M_h(j)$. Also assume the lowest point on $M_{h+1}(j)$ but not on $M_h(j)$ is lower than the lowest point on $M_{h+1}(j)$ but not on $M_h(j+1)$. Begining from the starting point, we let the points on $M_{h+1}(j)$ be up-points until we reach the top point of $M_{h+1}(j)$ or reach a point P_a which is a common point of $M_{h+1}(j)$ and a component chain of M_h, say $M_h(j-1)$.

Let the overlapped portion of $M_{h+1}(j)$ with $M_h(j-1)$ be denoted by $OV1$. According to Properties P1 and P3, there is an overlapped portion of $M_{h+1}(j-1)$ with $M_h(j-1)$, denoted by $OV2$, below $OV1$. If the points in $OV2$ have been processed, we follow $M_{h+1}(j)$ up until we reach another overlapped portion. Otherwise, we follow $M_h(j-1)$ down and make all points on $M_h(j-1)$ and between $OV1$ and $OV2$ be down-points; then, follow $M_{h+1}(j-1)$ up until we reach another overlapped portion.

Each time we meet an overlapped portion, we repeat the above procedure. Finally, we reach a top-point of M_{h+1} and obtain a z-chain Z. Obviously, the combination of Z and M_h is M_{h+1}. □

Based on Lemma 1, we need to obtain a maximum z-chain of a given maximum two-chain. We refer to the algorithm doing that as Algorithm MAX-THREE-CHAIN. This task is rather involved. However, the proposed algorithm employs mainly tree data structures (see [AK,LSL]). For details of Algorithm MAX-THREE-CHAIN and proofs see [LS], where the following theorem is proved.

Theorem 1 *Algorithm MAX-THREE-CHAIN finds a maximum three-chain of a point set* ρ *and runs in* $\Theta(n \log n)$ *time, where* $n = |\rho|$.

References

[AHU] Aho, A.V., Hopcroft, J.E., and Ullman, J.D., *The Design and Analysis of Computer Algorithms*, Addison Wesley, 1974.

[AK] Atallah, M.J., Kosaraju, S.R., "An Efficient Algorithm for Maxdominance, with Applications," *Algorithmica*, Vol. 4, No. 2, pp. 221-236, 1989.

[CL] Cong, J., and Liu, C.L., "On the k-Layer Planar Subset and Via Minimization Problems," manuscript, University of Illinois at Urbana-Champaign, 1989.

[Ga] Gavril, F., "Algorithms for Maximum k-colorings and k-coverings of Transitive Graphs" *Networks,* Vol. 17, 1987, pp. 465-470.

[LS] Lou, R.D., Sarrafzadeh, M., "Geometric Properties of Maximum k-chains", manuscript, Northwestern University, 1990.

[LSL] Lou, R.D., Sarrafzadeh, M., and Lee, D.T., "An Optimal Algorithm for he Maximum Two-chain Problem," In Proceedings of *First SIAM-ACM Conference on Discrete Algorithms (SODA),* San Francisco, CA, January 1990.

[PSh] Preparata, F.P. and Shamos, M.I., *Computational Geometry,* Springer-Verlag, New York, 1985.

[RKN] Rim, C.S., Kashiwabara, T., and Nakajima, K., "Exact Algorithms for Multilayer Topological Via Minimization", *IEEE Transactions on Computer-Aided Design*, Vol. 8, No. 11, pp. 1165-1184, November 1989.

[SLo] Sarrafzadeh, M., and Lou, R.D.,, " Maximum k-coverings of Weighted Transitive Graphs with Applications," To appear in *Algorithmica*, 1991.

[SLe1] Sarrafzadeh, M., and Lee, D.T., "A New Approach to Topological Via Minimization," *IEEE Transactions on Computer-Aided Design,* Vol. 8, No. 8, August 1989, pp. 890-900.

[SLe2] Sarrafzadeh, M., and Lee, D.T., "Topological Via Minimization Revisited," To appear in *IEEE Transactions on Computers*, 1991.

A Heuristic Algorithm for the k-Center Problem with Cost and Usage Weights[1]

Qingzhou Wang and Kam Hoi Cheng
Computer Science Department, University of Houston
Houston, Texas 77204-3475

Abstract

Given a graph where each vertex has both cost and usage weights, the problem is to choose vertices as service centers so that the maximum weighted service delivery distance to any vertex is minimized under the constraint that the total cost of construction is within k. The problem is NP-hard [2] and we propose an intuitive greedy strategy which first identify a vertex of highest usage, and then choose from its close by vertices the lowest cost vertex to build a service center. We have proved that our heuristic is a 3-approximation algorithm, i.e., it generates results (S_H) guaranteed to be no greater than three times the optimal solution values (S_O). We have also shown that unless P=NP, this is the best possible relative error heuristic, not only absolutely but also asymptotically, i.e., no polynomial time heuristic can achieve $S_H \leq \alpha \cdot S_O + \beta$ for $\alpha < 3$ and β a constant. Our problem is a generalization of six previously studied problems. For the *simple k-center problem* [1], [3], [4], the *k-center problem with usage weight* [6], and the *continuous k-center problem* [5], [6], 2-approximation algorithms have been proposed. For the *weighted k-center problem*, the *k-supplier problem*, and the *weighted k-supplier problem*, 3-approximation algorithms have been proposed [1], [4]. For those problems where 2-approximation heuristics exist, we have shown that our heuristic also achieves 2-approximation.

1. Introduction

Let $G = (V, E)$ be a weighted graph where a cost weight $\omega_c(v)$ and a usage weight $\omega_u(v)$ are associated with each vertex v, $v \in V$. For any two vertices u and v in V, the distance $\delta(u, v)$ is defined to be the length of the shortest path linking u and v, i.e., if the shortest path linking u and v is $e_1, e_2, \ldots, e_l$, then $\delta(u, v) = \sum_{i=1}^{l} |e_i|$ where $|e_i|$

---

[1] This research is based in part upon work supported by the Texas Advanced Research Program under grant 1028-ARP.

is the length of edge e_i. The *weighted* distance $\delta_\omega(u,v)$ from u to v is defined to be $\delta_\omega(u,v) = \delta(u,v) \cdot \omega_u(v)$. Note that $\delta_\omega(u,v)$ is generally not the same as $\delta_\omega(v,u)$. These definitions can easily be extended to a subset $S \subseteq V$: $\delta(S,v) = \min\{\delta(s,v) : s \in S\}$, $\delta_\omega(S,v) = \min\{\delta_\omega(s,v) : s \in S\}$. The k-center problem with cost and usage weights is to choose the vertex subset S, $S \subseteq V$, such that $\omega_c(S) = \sum_{s \in S} \omega_c(s) \leq k$ and the function $f(S) = \max\{\delta_\omega(S,v) : v \in V\}$ attains its minimum value, $r_\omega(G,k)$. The value $r_\omega(G,k)$ is called the *weighted service radius with cost* k of the graph G, i.e., $r_\omega(G,k) = \min_{\omega_c(S) \leq k} f(S)$.

The problem is NP-hard [2] and so we are interested in polynomial time approximation algorithms. An approximation algorithm is called an α-approximation algorithm if for all problem instances, it can guarantee that $\frac{S_H}{S_O} \leq \alpha$ where S_O and S_H are the optimal and the approximate solutions respectively. A heuristic satisfying $S_H \leq \alpha \cdot S_O + \beta$, where β is a constant is said to have an asymptotic performance α.

The k-center problem with cost and usage weights is the generalization of the following six previously studied problems.

1. *The simple k-center problem.* If $\forall v \in V$, $\omega_c(v) = \omega_u(v) = 1$, then our problem reduces to the simple k-center problem. Several 2-approximation algorithms have been proposed for the simple k-center problem [1], [3] and [4]. It is also shown in [4] that a 2-approximation algorithm is the best possible polynomial time α-approximation heuristic for this problem.

2. *The weighted k-center problem.* If $\forall v \in V$, $\omega_c(v) \in [0,\infty)$, $\omega_u(v) = 1$, then our problem reduces to the weighted k-center problem. Two 3-approximation algorithms have been proposed in [1] and [4] respectively. However, it is still an open problem whether a 3-approximation algorithm is the best possible heuristic with polynomial time.

3. *The k-center problem with vertex weight.* If $\forall v \in V$, $\omega_c(v) = 1$, $\omega_u(v) \in [0,\infty)$, then our problem reduces to the k-center problem with vertex weight. A 2-approximation algorithm have been developed in [6] which is also the best possible α-approximation heuristic with polynomial time.

4. *The continuous k-center problem.* This problem is similar to the simple k-center problem except that service centers may be chosen from any point along every edge $e \in E$. A polynomial time algorithm has been proposed in [5] to eliminate all but $O(|V|^4)$ points along these edges. The elimination process creates a new graph $G' = (V', E')$ where V'

is the union of V and the $O(|V|^4)$ new vertices along the edges in E, and E' consists of the edge pieces of E created by the new vertices in $V' - V$. By using the following vertex weight:

$$\begin{aligned} \omega_u(v) &= 1, \quad \text{if } v \in V, \\ \omega_u(v) &= 0, \quad \text{if } v \in V' - V, \end{aligned}$$

the original continuous k-center problem is equivalent to a k-center problem with vertex weight. Hence it is a special case of our problem, and a 2-approximation algorithm exists [6].

5. *The k-supplier problem.* Let $G = (V, E)$ where $V = V_s \cup V_c$. The sets V_s and V_c are called the supplier candidate set and the customer set respectively, and they may overlap. For any $S' \subseteq V_s$, the longest service distance through which any customer can be served by at least one supplier in S' is $f(S') = \max\{\delta(S', u) : \forall u \in V_c\}$. The k-supplier problem is to minimize the service distance to the customers in V_c under the constraint that no more than k suppliers are chosen. Formally, the objective is to attain $r_s(G, k) = \min_{|S'| \leq k} f(S')$. For any k-supplier problem, it is equivalent to the following k-center problem with cost and usage weights:

$$\begin{aligned} &\text{if } v \in V_s, \quad \omega_c(v) = 1, \quad \text{otherwise} \quad \omega_c(v) = k + 1; \\ &\text{if } v \in V_c, \quad \omega_u(v) = 1, \quad \text{otherwise} \quad \omega_u(v) = 0. \end{aligned}$$

The assignment of $k + 1$ to be the cost weight of all vertices not in V_s excludes them from being chosen as suppliers. Furthermore, with all vertices not in V_c being assigned an usage weight of 0, the weighted service distance to any non-customer vertex is zero.

6. *The weighted k-supplier problem.* This is a k-supplier problem where a cost weight, $\tilde{\omega}(v)$, is associated with each vertex in the supplier candidate set V_s. The objective is to minimize the longest service distance under the constraint that the total construction cost of the selected suppliers can not exceed k. Any weighted k-supplier problem can be transformed to the following k-center problem with cost and usage weights:

$$\begin{aligned} &\text{if } v \in V_s, \quad \omega_c(v) = \tilde{\omega}(v), \quad \text{otherwise} \quad \omega_c(v) = k + 1; \\ &\text{if } v \in V_c, \quad \omega_u(v) = 1, \quad \text{otherwise} \quad \omega_u(v) = 0. \end{aligned}$$

For both versions of the k-supplier problem, 3-approximation algorithms have been developed [1], [4], and 3-approximation is the best possible polynomial time α-approximation unless P=NP.

Throughout the paper, let $|V| = n$. We assume that both edge lengths and vertex weights are non-negative and the graph connected. So

$$0 \leq \delta(u,v) < \infty \quad \text{and} \quad 0 \leq \delta_\omega(u,v) < \infty, \quad \forall u,v \in V. \tag{1.1}$$

For a disconnected graph, we can first link different connected components with sufficiently long edges, and then apply our algorithm. Let m be the number of different weighted distances (d_i), then $m \leq n(n-1)$ and $d_0 < d_1 < \ldots < d_i < \ldots < d_{m-1}$. It is obvious that $\exists d_j$ such that $d_j = r_\omega(G,k)$, where $r_\omega(G,k)$ is the weighted radius of an optimal selection of k-centers on the graph G. Let d be a non-negative number. For a weighted graph G, the *d-restricted digraph* $G(d) = (V, E(d))$ has the same vertex set V as in G. The edge set $E(d)$ is defined as $E(d) = \{\vec{e}_{(u,v)} : \delta_\omega(u,v) \leq d\}$, where $\vec{e}_{(u,v)}$ is the directed edge in $G(d)$ from vertex u to vertex v. There are at most $m + 1 \leq n(n-1) + 1$ different d-restricted digraphs that can be derived from any graph G with n vertices.

2. A Polynomial Time Heuristic

We first present a greedy algorithm (Algorithm 1) to select a set of vertices each of which is the lowest construction cost neighbor of a vertex with the highest usage weight at the time of its selection. The input to the greedy algorithm is a d-restricted digraph $G(d) = (V, E(d))$. Let $u \in V$ be a vertex in $G(d)$. The *primary* neighbor vertex set of u, $N_1(u)$, is defined as

$$N_1(u) = \{u\} \cup \{v : \exists \vec{e}_{(v,u)} \in E(d)\}. \tag{2.1}$$

The *secondary* neighbor vertex set of u, $N_2(u)$, is defined as

$$N_2(u) = \{u\} \cup \{v : \exists \vec{e}_{(u,v)} \in E(d)\} \cup \{v : \exists x \in V, \ \vec{e}_{(x,u)}, \vec{e}_{(x,v)} \in E(d)\}. \tag{2.2}$$

Note that the order of vertices in the two definitions are important. The chosen set of vertices, $I_g(G(d))$, is called the *greedy* set. Note that in selecting a vertex from $N_1(u')$, if u' and some other vertices have a tie on the lowest cost, we will choose u'. We will call this the *tie-breaking rule* in future discussions. The preference of u' over other vertices during a tie is not crucial in most cases; however, in the simple, continuous and the usage weight k-center problems, such a selection criterion preserves their relative error ratio of 2. Algorithm 1 as well as the construction of a d-restricted digraph are easily seen to take polynomial time. Our proposed heuristic algorithm (Algorithm 2) applies

```
      input $d$-restricted digraph $G(d) = (V, E(d))$
      $\forall u \in V$, construct $N_1(u)$ and $N_2(u)$
      $I_g(G(d)) \leftarrow \emptyset$; $cost \leftarrow 0$
      repeat
(†)       select a $u' \in V$ with $\omega_u(u') = \max\{\omega_u(v) : v \in V\}$
(‡)       select a $u \in N_1(u')$ with $\omega_c(u) = \min\{\omega_c(v) : v \in N_1(u')\}$
          $cost \leftarrow cost + \omega_c(u)$; $I_g(G(d)) \leftarrow I_g(G(d)) \cup \{u\}$
          $V \leftarrow V - N_2(u')$
      until $V = \emptyset$
      return $(I_g(G(d)), cost)$
```

Algorithm 1. A Greedy Algorithm

Algorithm 1 to a sequence of d-restricted digraphs of the graph G. Since the **while** loop in Algorithm 2 executes at most $m \leq n(n-1)$ times and each iteration takes polynomial time, Algorithm 2 is easily seen to take polynomial time.

3. Analysis of the Heuristic Algorithm

First, we show the relationship among the primary neighbor vertex sets of those vertices processed by the (†) statement in Algorithm 1.

Theorem 1. If u_1' and u_2' are two vertices selected by the (†) statement in two different iterations of Algorithm 1, then $N_1(u_1') \cap N_1(u_2') = \emptyset$.

Proof: Since vertices u_1' and u_2' are selected in two different iterations by the (†) statement, let us assume that u_1' is selected before u_2'. This implies $\omega_u(u_1') \geq \omega_u(u_2')$.

If the assertion in the theorem is not true, then there is at least one vertex v' such that $v' \in N_1(u_1') \cap N_1(u_2')$. Since it is known that $u_1' \neq u_2'$, there are only three possible cases, and we will show that in all three cases, $u_2' \in N_2(u_1')$.

Case 1. $v' = u_1', v' \neq u_2'$. Since $u_1' = v' \in N_1(u_2')$, then there is a directed edge $\vec{e}_{(u_1', u_2')} \in E(d)$. From the definition in (2.2), we have $u_2' \in N_2(u_1')$.

Case 2. $v' \neq u_1', v' = u_2'$. Similar to *Case 1*, there is a directed edge $\vec{e}_{(u_2', u_1')} \in E(d)$. This implies $\delta_\omega(u_2', u_1') = \delta(u_2', u_1') \cdot \omega_u(u_1') \leq d$. Since $\omega_u(u_2') \leq \omega_u(u_1')$, so $\delta_\omega(u_1', u_2') =$

```
construct the sequence $d_0 < d_1 < \ldots < d_i < \ldots < d_{m-1}$
$i \leftarrow 0$; $DONE \leftarrow$ false
while $i < m$ and not $DONE$ do
    construct the $d$-restricted digraph $G(d_i)$
    use Algorithm 1 to get $I_g(G(d_i))$ and $cost_i$
    if $cost_i \leq k$ then
        $DONE \leftarrow$ true; $S \leftarrow I_g(G(d_i))$
        $COST \leftarrow cost_i$; $\tilde{d} \leftarrow d_i$
    endif
    $i \leftarrow i + 1$
endwhile
designate vertices in $S$ as service centers
```

Algorithm 2. The Heuristic Algorithm with Error Ratio 3

$\delta(u_1', u_2') \cdot \omega_u(u_2') \leq d$ and $\vec{e}_{(u_1', u_2')} \in E(d)$. Hence, $u_2' \in N_2(u_1')$.

Case 3. $v' \neq u_1', v' \neq u_2'$. Since $v' \in N_1(u_1') \cap N_1(u_2')$, then there are two directed edges originated from v', i.e., $\vec{e}_{(v', u_1')}, \vec{e}_{(v', u_2')} \in E(d)$. Once again, from the definition in (2.2), we have $u_2' \in N_2(u_1')$.

Since $u_2' \in N_2(u_1')$ and u_1' is selected first, u_2' will be deleted from V when u_1' is selected, so it contradicts the statement that u_2' is selected in a later iteration. Therefore, the theorem is proved. ■

From Theorem 1, no vertex will be selected by the (‡) statement more than once in Algorithm 1. This guarantees

$$cost = \sum_{u \in I_g(G(d))} \omega_c(u). \tag{3.1}$$

Furthermore, for each $u \in I_g(G(d))$, there is a unique u' selected by the (†) statement such that

$$\omega_c(u) = \min\{\omega_c(v) : v \in N_1(u')\}. \tag{3.2}$$

We call u' the *neighbor* of u, and denote it $n(u)$. This neighbor relationship, $u' = n(u)$, defines a one-to-one (but not onto) function from the greedy set $I_g(G(d))$ into V. For the simple k-center problem and the k-center problem with vertex weight, all vertices have the same cost weight, so by the tie-breaking rule in §2, we have

$$n(u) = u, \quad \forall u \in I_g(G(d)). \tag{3.3}$$

Theorem 2. When the **while** loop in Algorithm 2 terminates, $\tilde{d} \leq r_\omega(G, k)$.

Proof: Let $d_j = r_\omega(G, k)$ where $0 \leq j \leq m - 1$. Suppose Algorithm 1 is executed on $G(d_j)$, let l be the size of the greedy set $I_g(G(d_j))$. Let vertices in $I_g(G(d_j))$ be denoted as u_i, and $u'_i = n(u_i)$, $i = 1, \ldots, l$. Suppose the vertex u'_i is served by the vertex v_i under an optimal k-center arrangement, then $v_i \in N_1(u'_i)$ since $\delta(v_i, u'_i) \cdot \omega_u(u'_i) \leq d_j$. Since both $u_i \in N_1(u'_i)$ and $v_i \in N_1(u'_i)$, then $\omega_c(u_i) \leq \omega_c(v_i)$ by the statement (‡) in Algorithm 1. Furthermore, if $a \neq b$, $v_a \neq v_b$ since we know that $N_1(u'_a) \cap N_1(u'_b) = \emptyset$ from Theorem 1. Therefore,

$$cost_j = \sum_{u_i \in I_g(G(d_j))} \omega_c(u_i) \leq \sum_{i=1}^{l} \omega_c(v_i) \leq k. \tag{3.4}$$

As a result, the **while** loop in Algorithm 2 is guaranteed to stop no later than its j^{th} iteration, and hence must terminate after the j_0^{th} iteration with $j_0 \leq j$. Hence, $\tilde{d} = d_{j_0} \leq d_j = r_\omega(G, k)$. ■

Now suppose the **while** loop in Algorithm 2 terminates after the j_0^{th} iteration. Let $d_j = r_\omega(G, k)$ where $0 \leq j \leq m - 1$. Then from Theorem 2, we know that $j_0 \leq j$. Let l be the number of vertices in the greedy set and let them be represented by u_i, and $u'_i = n(u_i)$, $i = 1, \ldots, l$. We observe that during the execution of the greedy algorithm, the vertex set V is replaced repeatedly by $V - N_2(u'_i)$. Mathematically, this process can be represented by the following recurrence relations,

$$V_{i+1} = V_i - N_2(u'_i), \quad \text{with } \omega_u(u'_i) = \max_{v \in V_i} \omega_u(v), \quad i = 1, \ldots, l. \tag{3.5}$$

Note that it is a monotonically decreasing sequence of sets, i.e.,

$$V = V_1 \supset V_2 \supset \ldots \supset V_l \supset V_{l+1} = \emptyset; \tag{3.6}$$

and the set being removed from V_i is actually $N_2^{(i)}(u'_i)$ where

$$N_2^{(i)}(u'_i) = N_2(u'_i) \cap V_i, \quad i = 1, \ldots, l. \tag{3.7}$$

It is clear that both $N_2^{(i)}(u'_i) \subseteq V_i$ and $N_2^{(i)}(u'_i) \subseteq N_2(u'_i)$. It is also important to note that the construction of $N_1(u)$ and $N_2(u)$ for each u in Algorithm 1 is done before the

repeat loop because of the following scenario. The set $N_2^{(i)}(u_i')$ may contain less vertices than required if only vertices in V_i are considered. This is the result of the way we define the secondary neighbor vertex set N_2 in Eq. (2.2).

Theorem 3. Let u_i be the service center for all the vertices in $N_2^{(i)}(u_i')$, $i = 1, \ldots, l$. For any $v \in N_2^{(i)}(u_i')$, the weighted distance from u_i to v will not exceed $3\tilde{d}$, i.e., $\delta_\omega(u_i, v) \leq 3\tilde{d}$. For the simple, continuous and usage weight k-center problems, $\delta_\omega(u_i, v) \leq 2\tilde{d}$.

Proof: Since $N_2^{(i)}(u_i') \subseteq V_i$ and $\omega_u(u_i') = \max_{v \in V_i} \omega_u(v)$, we must have

$$\omega_u(u_i') \geq \omega_u(v), \quad \forall v \in N_2^{(i)}(u_i'). \tag{3.8}$$

From the definition of the secondary neighbor set in Eq. (2.2), v $(\neq u_i')$ is included in $N_2^{(i)}(u_i') \subseteq N_2(u_i')$ for one of the following two reasons:

$$v \in N_2^{(i)}(u_i'),\ v \neq u_i' \left\{ \begin{array}{ll} \text{reason 1:} & \exists \vec{e}_{(u_i', v)} \in E(\tilde{d}), \\ \text{reason 2:} & \exists x \in V, \text{ such that } \vec{e}_{(x, u_i')}, \vec{e}_{(x,v)} \in E(\tilde{d}) \end{array} \right. \tag{3.9}$$

where $E(\tilde{d})$ is the edge set of $G(\tilde{d}) = (V, E(\tilde{d}))$. If vertex v is included for the first reason, we have $\delta_\omega(u_i', v) \leq \tilde{d} \leq 2\tilde{d}$. Now, suppose vertex v is included for the second reason. The existence of $\vec{e}_{(x,u_i')}$ and $\vec{e}_{(x,v)}$ in $G(\tilde{d})$ implies

$$\begin{aligned} \delta_\omega(x, u_i') &= \delta(x, u_i') \cdot \omega_u(u_i') \leq \tilde{d}, \\ \delta_\omega(x, v) &= \delta(x, v) \cdot \omega_u(v) \leq \tilde{d}. \end{aligned} \tag{3.10}$$

We know that $\omega_u(u_i') \geq \omega_u(v)$ from Eq. (3.8), and $\delta(u_i', v) \leq \delta(x, u_i') + \delta(x, v)$ from the shortest path definition of δ in §1. These two facts combined with Eq. (3.10) imply

$$\delta_\omega(u_i', v) = \delta(u_i', v) \cdot \omega_u(v) \leq \delta(x, u_i') \cdot \omega_u(u_i') + \delta(x, v) \cdot \omega_u(v) \leq 2\tilde{d}. \tag{3.11}$$

Therefore, for any $v \in N_2^{(i)}(u_i')$, $\delta_\omega(u_i', v) \leq 2\tilde{d}$.

Next since $u_i \in N_1(u_i')$, so $\delta_\omega(u_i, u_i') = \delta(u_i, u_i') \cdot \omega_u(u_i') \leq \tilde{d}$. Again we know that $\omega_u(u_i') \geq \omega_u(v)$ from Eq.(3.8), and $\delta(u_i, v) \leq \delta(u_i, u_i') + \delta(u_i', v)$ from the shortest path definition of δ in §1. These two facts combined with Eq.(3.11) imply

$$\begin{aligned} \delta_\omega(u_i, v) &= \delta(u_i, v) \cdot \omega_u(v) \\ &\leq \delta(u_i, u_i') \cdot \omega_u(u_i') + \delta(u_i', v) \cdot \omega_u(v) \\ &\leq \tilde{d} + 2\tilde{d} = 3\tilde{d}. \end{aligned} \tag{3.12}$$

Therefore, we can conclude that for any $v \in N_2^{(i)}(u_i')$, $\delta_\omega(u_i, v) \le 3\tilde{d}$. The proof of the second statement is similar. ■

Theorem 4. Let the set S produced in Algorithm 2 be the chosen set of k-centers. Let $r_{\omega,H}(G,k)$ be the maximum weighted distance from S to any vertex in V, then

$$\frac{r_{\omega,H}(G,k)}{r_\omega(G,k)} \le 3. \tag{3.13}$$

In other words, Algorithm 2 is a 3-approximation algorithm. It is a 2-approximation algorithm for the simple, continuous and usage weight k-center problems.

Proof: From Algorithm 2, $S = I_g(G(\tilde{d}))$. In Theorem 3, we let vertex $u_i \in I_g(G(\tilde{d}))$ to serve all vertices in $N_2^{(i)}(u_i')$, Since $V = \cup_{i=1}^{l} N_2^{(i)}(u_i')$, so every vertex in V is served by a vertex in S. In addition, from the terminating condition in Algorithm 2, we know that $\sum_{u_i \in I_g(G(\tilde{d}))} \omega_c(u_i) \le k$. Therefore, the total construction cost at all vertices of S is within the constraint, k. Finally, for any $v \in V$, $\delta_\omega(I_g(G(\tilde{d})), v) \le 3\tilde{d}$. Hence,

$$\begin{aligned} r_{\omega,H}(G,k) &= \max\{\delta_\omega(S,v) : v \in V\} = \max\{\delta_\omega(I_g(G(\tilde{d})), v) : v \in V\} \\ &\le 3\tilde{d} \le 3d_j = 3r_\omega(G,k). \end{aligned}$$

The proof for 2-approximation is similar. ■

It is obvious that our 3-approximation heuristic is the best possible polynomial time α-approximation algorithm since 3 is the best possible polynomial time α-approximation algorithm for the k-supplier problem [4]. The next theorem shows that unless P=NP, it is impossible to design a polynomial time heuristic which achieves an asymptotic performance ratio better than 3.

Theorem 5. A polynomial time approximation algorithm of the k-supplier problem with asymptotic performance ratio $\alpha < 3$ can solve the NP-complete hitting set problem in polynomial time.

Proof: First we define the hitting set problem. Let H be a finite set, and H_i, $i = 1, \ldots, m$ be m subsets of H. A *hitting set* H' is a subset of H such that $H' \cap H_i \ne \emptyset$, for all i. The hitting set problem is to determine if there is a hitting set of size k, and it is NP-complete [2].

Suppose an approximation algorithm has the following asymptotic performance ratio

$$S_H \le \alpha \cdot S_O + \beta, \quad \text{with } \alpha < 3 \text{ and } \beta > 0, \tag{3.14}$$

where S_O and S_H are the optimal and the approximate solutions respectively. For a given instance of the hitting set problem, $H = \{u_1, \ldots, u_l\}$ and $H_i \subseteq H$, $i = 1, \ldots, m$, we create two subsets of vertices

$$\begin{array}{lrcl} \text{customers:} & V_c & = & \{u_j : j = 1, \ldots, l\}, \\ \text{supplier candidates:} & V_s & = & \{v_{H_i} : i = 1, \ldots, m\}. \end{array} \tag{3.15}$$

A complete graph, $G = (V, E)$, is created with $V = V_s \cup V_c$ as the vertex set, and edge lengths are defined as follows:

$$\begin{array}{rcll} |e_{(u_j,u_{j'})}| & = & 2\eta, & \forall u_j, u_{j'} \in V_c, \\ |e_{(v_{H_i},v_{H_{i'}})}| & = & 2\eta, & \forall v_{H_i}, v_{H_{i'}} \in V_s, \\ |e_{(u_j,v_{H_i})}| & = & \eta, & \text{if } u_j \in H_i, \\ |e_{(u_j,v_{H_i})}| & = & 3\eta, & \text{if } u_j \notin H_i, \end{array} \tag{3.16}$$

where $\eta \in (0, \frac{\beta}{3-\alpha})$. The optimal service radius $r_s(G,k)$ can assume only two possible values, η or 3η. Let S be the set of k suppliers obtained heuristically. If $r_s(G,k) = \eta$, we must have

$$\max_{v \in V_c} \delta(S, v) \leq \alpha \cdot \eta + \beta < 3\eta, \text{ since } \eta \in (0, \frac{\beta}{3-\alpha}). \tag{3.17}$$

Otherwise, we must have

$$\max_{v \in V_c} \delta(S, v) \geq r_s(G,k) = 3\eta. \tag{3.18}$$

It is obvious that $r_s(G,k) = \eta$ if and only if there is a hitting set of size k, so $\max_{v \in V_c} \delta(S,v) < 3\eta$ if and only if there is a hitting set of size k. ■

4. References

[1] T. Feder and D. H. Greene, Optimal Algorithms for Approximate Clustering, *ACM Symposium on Theory of Computing*, 1988, pp. 434–444.

[2] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, *Freeman, San Francisco*, 1979.

[3] T. F. González, Clustering to Minimize the Maximum Intercluster Distance, *Theoretical Computer Science*, Vol. 38, 1985, pp. 293–306.

[4] D. S. Hochbaum and D. B. Shmoys, A Unified Approach to Approximation Algorithms for Bottleneck Problems, *Journal of ACM*, Vol. 33, 1986, pp. 533–550.

[5] E. Minieka, The m-Center Problem, *SIAM Review*, Vol. 12, 1970, pp. 138–139.

[6] J. Plesnik, A Heuristic for the p-Center Problem in Graphs, *Discrete Applied Mathematics*, Vol. 17, 1987, pp. 263–268.

Mean-Square-Error Analysis of ESPRIT under Random Sensor Uncertainties.

V. C. Soon and Y. F. Huang

Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556.

Abstract

The effects of random sensor uncertainties on the mean-square-error (MSE) of the ESPRIT algorithm for estimating direction-of-arrival (DOA) angles of multiple, narrowband, far-field signals is investigated. The random sensor uncertainties considered are those due to sensor gain and phase errors, sensor location errors and sensor pair misalignments (due to rotation errors). A general formulation of these various types of sensor uncertainties is presented. Expressions of the MSE of the DOA estimates under these perturbations are tgiven. Closed form solutions for the one source case and bounds on the MSE for the two source case are given. Our results indicate that the performance of ESPRIT under sensor perturbations can be improved by increasing the number of sensors in the sub-arrays. Sensor phase errors are found to be more critical than gain errors in the one and two source problems. Simulations are presented tand confirm the analysis results.

I. INTRODUCTION

In recent years, eigenstructure-based sensor array signal processing techniques have attracted a great deal of interest especially in radar and sonar signal processing. Eigenstructure-based methods such as MUSIC and ESPRIT [1,2] have been proposed for these applications, with a lot of attention being paid to the Direction-of-Arrival (DOA) estimation problem for narrowband, far field sources.

This paper presents an analysis of the mean-squared error (MSE) of ESPRIT DOA angle estimates under random perturbations of the sensors in the array, specifically, under random variations in sensor gains, phases, locations and rotational alignments of the sensor pairs. Both MUSIC and ESPRIT work very well when the sensors in the array are either calibrated (i.e. array manifold is known for MUSIC) or are matched (for ESPRIT). However, in practice, it may not always be possible to know the precise gains and phases or locations of the sensors in the array or to have sensor pairs that are perfectly aligned. Such model deviations may be due to environmental effects on the array. In [3], an analysis was presented for the Subspace Fitting algorithms (which include ESPRIT and MUSIC, among others) under sensor errors, where the steering matrix is assumed to be perturbed in a random manner. However, there was no direct dependence between the random perturbations in the sensor parameters such as gains, phases, locations or sensor pair alignments and the random perturbations assumed in the steering matrix in [3]. A deterministic analysis of the one source case for ESPRIT can be found in [4].

The mean-square-error expressions for ESPRIT under statistical variations of sensor parameters (such as the gains, phases and locations and rotations) is presented. The different types of perturbations can be modeled in a single general model under the assumption of small perturbations. The MSE of the DOA estimates are shown to be proportional to the variances of the sensor parameter perturbations, i.e., the variance of the sensor gains and phases perturbations, sensor location perturbations and sensor misalignments due to rotational perturbations. The MSE for the special cases of the one and two source DOA problem will be examined in greater detail. *In this paper we will be concerned with the asymptotic case, where the perturbed covariance is assumed known (i.e. finite snapshots effects are not investigated here).*

The problem formulation is presented in Section II, where the nominal array problem is stated and where the effects of small perturbations in the sensor gains, phases, locations and rotations of sensor pairs on the observations of the ESPRIT X and Y subarrays are presented. Section III contains statistical analysis results on the MSE of the ESPRIT DOA estimates under these sensor array perturbations. Simulations confirming our analysis are presented in Section IV and the results in the paper are summarized in Section V.

II. PROBLEM FORMULATION

Throughout this paper, it will be assumed that there are n narrowband, far-field sources at DOA angles $\{\theta_k\}$, k=1,2, ..., n, relative to the normal of the sensor pairs under nominal conditions, and m sensors in each of the ESPRIT X and Y sub-arrays at locations $\{\mu_i,\upsilon_i\}$ and $\{\bar{\mu}_i,\bar{\upsilon}_i\}$ respectively. Nominally, the paired sensors in the X and Y sub-arrays are perfectly matched, with a distance d between them [2]. Fig. 1 shows an example of the ESPRIT X and Y sub-arrays. By choosing the (μ,υ) axis appropriately, the sensor locations of the X and Y sub-arrays can be written as (μ_i,υ_i) and (μ_i+d,υ_i) respectively. The signal covariance matrix $Q=E\{ss^H\}$ is assumed to be positive definite. Then the observation vectors from the X and Y sub-arrays can be written as:

$$\text{X sub-array: } x = \Gamma A s + \eta_x \qquad (1.a)$$

$$\text{Y sub-array: } y = \Gamma A \Phi s + \eta_y \qquad (1.b)$$

where A is the steering matrix for the X sub-array, s is the signal vector while η_x and η_y are the additive uncorrelated noise vectors with zero means and $\sigma^2 I$ covariances. Specifically,

$A = [\ a_1 |\ a_2 |. ..|\ a_n\ ]$ with

$a_k = [\exp(-j2\pi(\upsilon_1\cos\theta_k+\mu_1\sin\theta_k)/\lambda)\ \ldots\ldots\ \exp(-j2\pi(\upsilon_m\cos\theta_k+\mu_m\sin\theta_k)/\lambda)]^T$

$\Gamma = \mathrm{diag}\{\alpha_1\exp(j\phi_1)\ \alpha_2\exp(j\phi_2)\ \ldots\ \alpha_m\exp(j\phi_m)\}$, a diagonal array of nominal sensor gains $\{\alpha_k\}$ and phases $\{\phi_k\}$.

$\Phi = \mathrm{diag}\{\exp(-j2\pi d\sin\theta_1/\lambda), \exp(-j2\pi d\sin\theta_2/\lambda), \ldots, \exp(-j2\pi d\sin\theta_n/\lambda)\}$

ΓA is assumed to have full column rank, i.e, rank(ΓA) = n. Throughout this paper, the superscripts "H" denotes the Hermitian , "T" denotes the transpose, "*" denotes the complex conjugate and $(.)^\#$ denotes the Moore-Penrose pseudo-inverse of the term within the brackets.

Under *small perturbations*, (see [5] for further details) the perturbed array signal processing problems can be formulated by the following general model :

$$x=(\Gamma A+\partial A_x)s+\eta_x \tag{2.a}$$

$$y=(\Gamma A+\partial A_y)\Phi s+\eta_y \tag{2.b}$$

with the matrix perturbation terms ∂A_x and ∂A_y being defined according to whether the perturbation is in terms of sensor gains and phases, sensor locations or sensor pair rotation errors.

II.1 Perturbation in sensor gains and phases

Assume that the sensor gains and phases in both X and Y subarrays are perturbed. Then

$$\partial A_x = \Delta\Gamma_1 A \text{ and } \partial A_y = \Delta\Gamma_2 A \tag{3}$$

where

$$\Delta\Gamma_1 = \mathrm{diag}\{(\Delta\alpha_1 + j\alpha_1\Delta\phi_1)\exp(j\phi_1), \ldots, (\Delta\alpha_m + j\alpha_m\Delta\phi_m)\exp(j\phi_m)\}$$

and

$$\Delta\Gamma_2 = \mathrm{diag}\{(\Delta\bar{\alpha}_1 + j\alpha_1\Delta\bar{\phi}_1)\exp(j\phi_1), \ldots, (\Delta\bar{\alpha}_m + j\alpha_m\Delta\bar{\phi}_m)\exp(j\phi_m)\}.$$

Here, the $(\Delta\alpha_i,\Delta\phi_i)$ and $(\Delta\bar{\alpha}_i,\Delta\bar{\phi}_i)$ terms denote the sensor gain and phase perturbations of the X and Y sub-array sensors respectively.

II.2 Perturbations in sensor locations.

Assume that the locations of sensors in both sub-arrays are perturbed.Writing the perturbed observations in matrix vector form, we have

where

$$\partial A_x = \Gamma\Delta A_x \text{ and } \partial A_y = \Gamma\Delta A_y \tag{4}$$

$$\Delta A_x = -j\frac{2\pi}{\lambda}\left\{\begin{bmatrix} \Delta\mu_1\, 0 \ldots\ldots 0 \\ 0\ \Delta\mu_2\, 0..0 \\ 0\ \ldots\ldots\ldots 0 \\ 0\quad \ldots\Delta\mu_m \end{bmatrix} A_s + \begin{bmatrix} \Delta\upsilon_1\, 0 \ldots\ldots 0 \\ 0\ \Delta\upsilon_2\, 0..0 \\ 0 \ldots\ldots\ldots 0 \\ 0\quad ..\Delta\upsilon_m \end{bmatrix} A_c\right\} \tag{5.a}$$

$$\Delta A_y = -j\frac{2\pi}{\lambda}\left\{\begin{bmatrix} \Delta\hat{\mu}_1\, 0 \ldots\ldots 0 \\ 0\ \Delta\hat{\mu}_2\, 0..0 \\ 0\qquad ...0 \\ 0\quad ..\Delta\hat{\mu}_m \end{bmatrix} A_s + \begin{bmatrix} \Delta\hat{\upsilon}_1\, 0 \ldots\ldots 0 \\ 0\ \Delta\hat{\upsilon}_2\, 0..0 \\ 0\qquad ...0 \\ 0\quad ..\Delta\hat{\upsilon}_m \end{bmatrix} A_c\right\} \tag{5.b}$$

with

$$A_s=[\sin\theta_1 a_1|..|\sin\theta_n a_n] \text{ and } A_c=[\cos\theta_1 a_1|..|\cos\theta_n a_n]. \tag{6}$$

Here, the $(\Delta\mu_i,\Delta\upsilon_i)$ and $(\Delta\hat{\mu}_i,\Delta\hat{\upsilon}_i)$ terms denote the sensor gain and phase perturbations of the X and Y sub-array sensors respectively.

II.3 Rotation of sensor pairs (doublets) in X and Y sub-arrays

In this section, we consider the case where the locations of the sensor pairs are perturbed by small angle rotations, causing the sensor pairs not to be aligned in the same direction. In particular, the distance between the sensors in each sensor pair remains the same, but their orientations are perturbed by small angle rotations around the mid-point between the sensors in the sensor pairs as shown in Fig. 2.

Let the small angle rotations of the sensor pairs be denoted by $\{\beta_k\}$, k=1, 2, ..., m. Then, using similar small perturbation approximations as in the previous sections, it can be shown that

$$\partial A_x = \Gamma \Delta A_x \text{ and } \partial A_y = \Gamma \Delta A_y \tag{7}$$

where

$$\Delta A_x = -j\frac{\pi d}{\lambda}\begin{bmatrix} \beta_1 0 \ldots 0 \\ 0\, \beta_2 \ldots .0 \\ \ldots \\ 0 \ldots\ldots \beta_m \end{bmatrix} A_c \text{ and } \Delta A_y = +j\frac{\pi d}{\lambda}\begin{bmatrix} \beta_1 0 \ldots 0 \\ 0\, \beta_2 \ldots .0 \\ \ldots \\ 0 \ldots\ldots \beta_m \end{bmatrix} A_c \tag{8}$$

with A_c defined as in Eq. (6).

III. ANALYSIS OF ESPRIT UNDER SMALL ARRAY PERTURBATIONS

Under the general perturbed model of Eqs. (2), application of the ESPRIT algorithm will result in the perturbation of the k-th DOA angle estimate, such that $\hat{\theta}_k = \theta_k + \Delta\theta_k$ from the true DOA angle θ_k, k =1, 2, ..., n.

We present the following results on the mean-square-error of the ESPRIT estimates under the different perturbation scenarios. Details of the proofs are given in [5].

Theorem 1 (Random perturbations of sensor gains and phases)

Assume that the sensor arrays are perturbed only in the sensor gains and phases with the following statistical properties:

(A.1.1) The sensor gain perturbations have zero means and are uncorrelated with each other, with variance σ_α^2.

(A.1.2) The sensor phase perturbations have zero means and are uncorrelated with each other, with variance σ_ϕ^2.

(A.1.3) The sensor gain perturbations are uncorrelated with the phase perturbations.

Then, the ESPRIT DOA angle estimates have the following MSE:

$$E\{|\Delta\theta_k|^2\} = \left(\frac{\lambda}{2\pi d\cos\theta_k}\right)^2 \left\{ b_k^T D_k G_1 D_k^H b_k^* - \mathrm{Re}\left(b_k^T D_k G_2 D_k b_k\right)\right\} \tag{10}$$

where $G_1 = \begin{bmatrix} (\sigma_\alpha^2 + \sigma_\phi^2 \alpha_2^2)\ 0 \ldots 0 \\ \ldots \\ 0 \ldots 0\ (\sigma_\alpha^2 + \sigma_\phi^2 \alpha_m^2) \end{bmatrix}$ and $G_2 = \begin{bmatrix} e^{j2\phi_1}(\sigma_\alpha^2 - \sigma_\phi^2 \alpha_1^2)\ 0 \ldots 0 \\ \ldots \\ 0 \ldots 0\ e^{j2\phi_m}(\sigma_\alpha^2 - \sigma_\phi^2 \alpha_m^2) \end{bmatrix}$

with b_k^T being defined from $(\Gamma A)^{\#}$ as follows

$$(\Gamma A)^{\#} = \begin{bmatrix} b_1^T \\ b_2^T \\ \cdot \\ \cdot \\ b_n^T \end{bmatrix} \tag{11}$$

and $D_k = \mathrm{diag}\{a_k\}$, an mxm diagonal matrix formed from the components of the vector a_k, with the (i,i)-th element of the matrix corresponding to the i-th element of the vector a_k.

Remark: When the nominal sensor gain and phase diagonal matrix is identity (i.e. unit sensor gains and zero phases), the MSE of the ESPRIT DOA estimates attain the simpler form of

$$E\{|\Delta\theta_k|^2\} = \left(\frac{\lambda}{2\pi d\cos\theta_k}\right)^2 \left\{ (\sigma_\alpha^2+\sigma_\phi^2)|b_k|^2 - (\sigma_\alpha^2-\sigma_\phi^2)\mathrm{Re}(b_k^T D_k^2 b_k) \right\} \tag{12.a}$$

Using this equation, bounds for the MSE are easily shown to be

$$2\left(\frac{\lambda}{2\pi d\cos\theta_k}\right)^2 \min\{\sigma_\alpha^2, \sigma_\phi^2\}|b_k|^2 \le E\{|\Delta\theta_k|^2\} \le 2\left(\frac{\lambda}{2\pi d\cos\theta_k}\right)^2 \max\{\sigma_\alpha^2, \sigma_\phi^2\}|b_k|^2 \tag{12.b}$$

Theorem 2 (Random perturbations in sensor locations)

Assume that the sensor arrays are perturbed only in the sensor locations in the (υ,ν)-axis with the following statistical properties:

(A.2.1) The μ–axis sensor location perturbations have zero means and are uncorrelated with each other, with variance σ_μ^2.

(A.2.2) The υ–axis sensor location perturbations have zero means and are uncorrelated with each other, with variance σ_υ^2.

(A.2.3) The μ–axis sensor location perturbations are uncorrelated with the υ–axis location perturbations.

Then, the ESPRIT DOA angle estimates have the following MSE:

$$E\{|\Delta\theta_k|^2\} = \left(\frac{1}{\cos\theta_k}\right)^2 \left(\left(\frac{\sigma_\mu}{d}\right)^2 \sin^2\theta_k + \left(\frac{\sigma_\upsilon}{d}\right)^2 \cos^2\theta_k \right) \left\{ |c|_k^2 + \mathrm{Re}\left(c_k^T D_k^2 c_k\right) \right\} \tag{13}$$

where c_k^T is defined as $c_k^T = b_k^T \Gamma$ where b_k^T is defined in Eq. (11) and D_k is defined as in Theorem 1.

Remark: From Eq. 13, we see that the ratios of the location error variances and the distance between sensor-pair, d are important in affecting MSE. If the error variances are small relative to d, then the corresponding MSEs are also small.

Theorem 3 (Random rotations of sensor pairs)

Assume that the sensor pairs in the arrays are perturbed only by random small angle rotations $\{\beta_k\}$, k=1, .., m, with the following statistical property:

(A.3.1) The random angle rotations $\{\beta_k\}$ have zero means and are uncorrelated with each other, with variance σ_β^2.

Then, the ESPRIT DOA angle estimates have the following MSE:

$$E\{|\Delta\theta_k|^2\} = \frac{\sigma_\beta^2}{2}\left\{|c|_k^2 + \mathrm{Re}\left(c_k^T D_k^2 c_k\right)\right\} \tag{14}$$

where c_k^T and D_k are as defined previously.

One and Two Source Cases

The MSE expressions of the ESPRIT estimates under sensor perturbations will now be applied to the special cases of the one and two source problems. The investigation of these special cases reveals an interesting result in that the robustness of the ESPRIT algorithm against sensor perturbations can be improved by increasing the number of sensors used in the sub-array. Also, it will be seen that in the case of sensor gains and phase perturbations, the sensor phase errors are more critical in terms of their effect on MSE than sensor gain errors.

(A). One Source Case.

Assume that Γ is an identity matrix and a single source is present.

Corollary 1 (One Source Case)

(I.1) Under random perturbations of sensor gains and phases, the MSE of the ESPRIT DOA angle estimate is found to be

$$E\{|\Delta\theta_1|^2\} = \frac{2\sigma_\phi^2}{m}\left(\frac{\lambda}{2\pi d\cos\theta_1}\right)^2 \tag{15}$$

(I.2) Under random perturbations of sensor locations, the MSE of the ESPRIT DOA angle estimate is found to be

$$E\{|\Delta\theta_1|^2\} = \frac{2}{m}\left(\frac{1}{\cos\theta_1}\right)^2\left(\left(\frac{\sigma_\mu}{d}\right)^2\sin^2\theta_1 + \left(\frac{\sigma_\upsilon}{d}\right)^2\cos^2\theta_1\right) \tag{16}$$

(I.3) Under random small angle rotations of the sensor pairs, the MSE of the ESPRIT DOA estimate is found to be

$$E\{|\Delta\theta_1|^2\} = \frac{\sigma_\beta^2}{m} \tag{17}$$

Remarks:

1. The MSE for the one source problem is inversely proportional to m. Hence, increasing the number of sensors in the sub-arrays will improve performance.
2. Part (I.1) states that under sensor gains and phase perturbations, the MSE is dependent only on sensor phase perturbations and not on that of the gains.
3. Part (I.3) states the interesting result that under sensor rotation errors, the MSE is independent of the true DOA of the source.

(B). Two Source Case

Again assume that Γ is an identity matrix and that there are two signal sources present. Let $\varsigma = a_1^H a_2$. Note that $|\varsigma|^2 \le m^2$. Then it can be shown that

$$|b_1|^2 = |b_2|^2 = \frac{1/m}{\left(1 - \frac{|\varsigma|^2}{m^2}\right)} \quad \text{and} \quad b_1^T D_1^2 b_1 = \left(b_2^T D_2^2 b_2\right)^* = \frac{1/m}{\left(1 - \frac{|\varsigma|^2}{m^2}\right)^2}\left\{1 - 2\frac{|\varsigma|^2}{m^2} + \frac{1}{m^3}\varsigma^2 a_2^H D_1^2 a_2^*\right\}$$

Using these expressions and Theorems 1 - 3 we obtain

Corollary 2 (Two Source Case)

(II.1) Under random perturbations of sensor gains and phases, the MSE of the ESPRIT DOA angle estimates are found to be

$$\left(\frac{\lambda}{2\pi d\cos\theta_k}\right)^2 \min\{\sigma_\alpha^2, \sigma_\phi^2\}\frac{2/m}{(1 - \frac{|\varsigma|^2}{m^2})} \le E\{|\Delta\theta_k|^2\} \le \left(\frac{\lambda}{2\pi d\cos\theta_k}\right)^2 \max\{\sigma_\alpha^2, \sigma_\phi^2\}\frac{2/m}{(1 - \frac{|\varsigma|^2}{m^2})}\ ; k = 1, 2. \quad (18)$$

(II.2) Under random perturbations of sensor locations, the MSE of the ESPRIT DOA angle estimates are found to be

$$E\{|\Delta\theta_k|^2\} \le \left(\frac{1}{\cos\theta_k}\right)^2\left(\left(\frac{\sigma_\mu}{d}\right)^2 \sin^2\theta_k + \left(\frac{\sigma_\upsilon}{d}\right)^2 \cos^2\theta_k\right)\frac{2/m}{(1 - \frac{|\varsigma|^2}{m^2})}\ ; k = 1, 2. \quad (19)$$

(II.3) Under random small angle rotations of the sensor pairs, the MSE of the ESPRIT DOA estimates are found to be

$$E\{|\Delta\theta_k|^2\} \le \sigma_\beta^2\frac{1/m}{(1 - \frac{|\varsigma|^2}{m^2})}\ ; k = 1, 2. \quad (20)$$

Remarks:

1. From the expressions in Eqs. (18-20), we can see that when $|\varsigma|^2/m^2 < 1$(strictly less than unity), then the MSE of the ESPRIT DOA estimates will improve as the number of sensors gets larger. So, at least for the one and two source cases, the robustness of the ESPRIT algorithm against sensor perturbations can be improved by increasing the number of sensors in the sub-arrays.n. For the special case of a uniform linear array see [5].

2. For large m, the expressions of $|b_k|^2$ and Real$\{b_k^T D_k^2 b_k\}$ are approximately equal to 1/m. Using this approximation in Theorem 1 (for sensor gain and phase perturbations) we note that the sensor gain variance term cancels. Hence, the approximate solution for the sensor gain and phase perturbation case is :

$$E\{|\Delta\theta_k|^2\} \approx \left(\frac{\lambda}{2\pi d\cos\theta_k}\right)^2 \frac{2\sigma_\phi^2}{m}\ ; k = 1, 2. \quad (21)$$

Therefore, for large m, the MSE of ESPRIT DOA estimates are dependent only on the sensor phase perturbations.

IV. SIMULATIONS

In this section, the nominal sensor gain and phase values are assumed to be unity and zero respectively, i.e., $\Gamma = I$.

Simulation : (Three Source Case)

In this example, there are three uncorrelated signal sources at the DOA angles of -20^o, -10^o and $+15^o$ where the first two signals are of unit power while the third is at $(.8)^2$ power. The size of the sub-arrays is $m=4$ with the nominal distance between sensors in a sensor pair $d=\lambda/2$. The sensor positions of the X and Y sub-arrays are $\{(-\lambda/4,-\lambda/2),(+\lambda/4,0),(-\lambda/4,+\lambda/2),(-3\lambda/4,0)\}$ and $\{(+\lambda/4,-\lambda/2),(+3\lambda/4,0),(+\lambda/4,+\lambda/2),(-\lambda/4,0)\}$ respectively.. One thousand Monte Carlo trials are conducted to obtain the averaged TLS-ESPRIT DOA MSE. In Fig. 3, the lines will denote the theoretical MSE as predicted by our analysis and the asterisk "*" symbols will denote the TLS-ESPRIT MSE. Fig. 3 shows the case where the sensor gains and phases were randomly perturbed.

There is a good agreement between the theoretical MSE of ESPRIT DOA angle estimates for the array perturbation as stated in Theorems 1 and the simulation results.

V. CONCLUSIONS

The MSE of the ESPRIT method for direction of arrival (DOA) estimation of multiple narrowband, far-field signals under small array perturbations is analysed here. Specifically, assuming that the observation covariance matrix is known (i.e. finite sample effects were not considered), the effects of random perturbations in the sensor gains and phases, random perturbations in sensor locations and random rotation errors of sensor pairs on the performance of ESPRIT are investigated. Under small perturbations, the formulations of the perturbed array problem with the above mentioned perturbations is shown to be incorporated into a single general perturbation model. The expressions for the MSE of the ESPRIT DOA estimates were derived. The special cases of the one and two source problems were investigated in greater detail. The analysis results indicate that the MSE will decrease as more sensors are used in the ESPRIT sub-arrays and that sensor phase perturbations are more important in affecting MSE than sensor gain perturbations. Computer simulations were conducted and it was found that the simulation MSE are well predicted by our analytical expressions.

References

[1] R. O. Schmidt, "Multiple Emitter Location and Signal Parameters Estimation," *IEEE Trans. Antenna. Prop.*, vol. AP-34, pp.276-280, Mar. 1986.

[2] R. Roy and T. Kailath, "ESPRIT-Estimation of Signal Parameters via Rotational Invariance Techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 984-993, July 1989.

[3] A. L. Swindlehurst and T. Kailath, "An Analysis of the Subspace Fitting Algorithms in the Presence of Sensor Errors," *Proc. IEEE Int'l. Conf. Acoust., Speech, Signal Processing* , Alburqueque, NM, pp. 2647-2650, 1990.

[4] E. E. Hines, "ESPRIT with perturbed subarrays," *Proc. IEEE Int'l. Conf. Acoust., Speech, Signal Processing* , Alburqueque, NM, pp. 2947-2950, 1990.

[5] V.C. Soon and Y. F. Huang, "Analysis of ESPRIT under Random Sensor Uncertainties," Technical Report #LISA 90-01, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556, August 1990.

Acknowledgements: The authors would like to thank Dr. R. Liu and Mr. L. Tong, both of the department of Electrical Engineering, University of Notre Dame, for the stimulating discussions held in the course of this research.

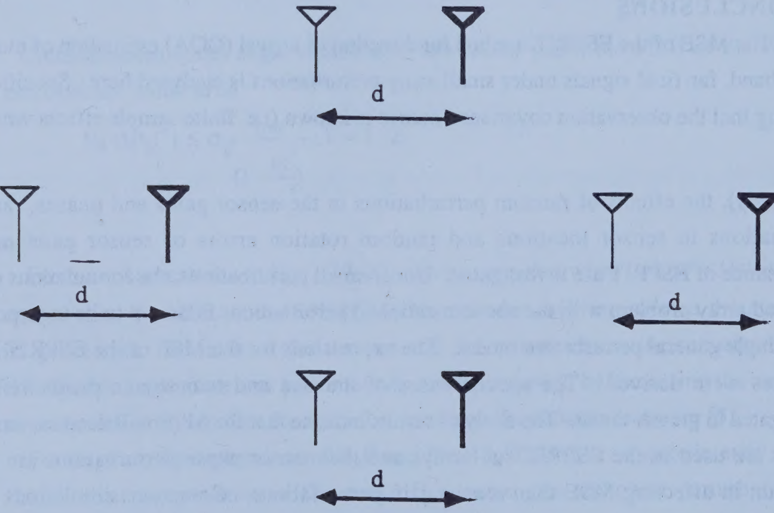


Figure 1: Example of ESPRIT X and Y sub-arrays.

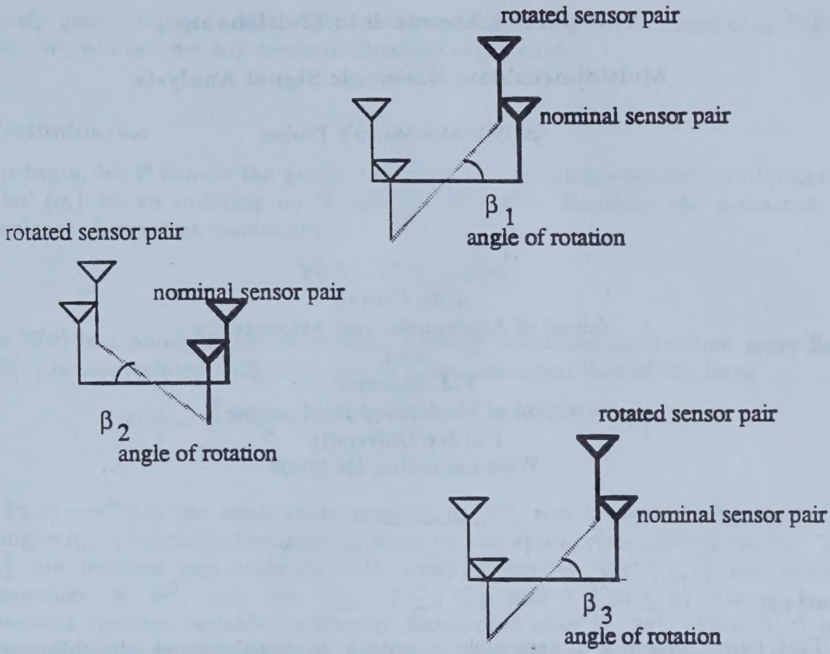

rotated sensor pair
nominal sensor pair
β_1
angle of rotation
rotated sensor pair
nominal sensor pair
β_2
angle of rotation
rotated sensor pair
nominal sensor pair
β_3
angle of rotation

Figure 2: An example of misaligned sensor-pairs due to rotational errors.

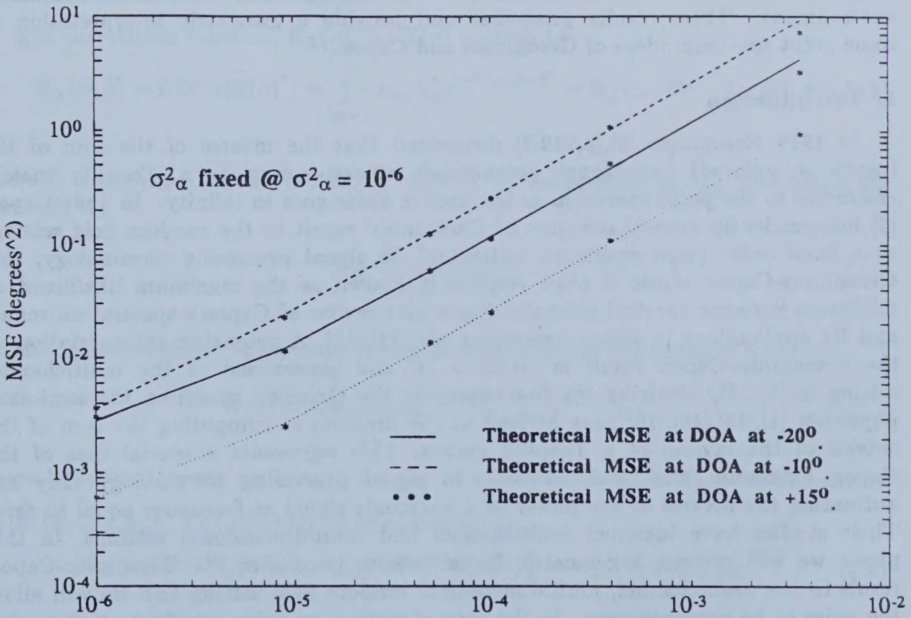


Figure 3: MSE with varying σ_ϕ^2 and fixed σ_α^2 of three source problem of Simulation 2.

A Geometric Approach to Multichannel, Multidimensional Harmonic Signal Analysis in Nonstationary Noise

by

A. E. Frazho
School of Aeronautics and Astronautics
and
P.J. Sherman
School of Mechanical Engineering
Purdue University
West Lafayette, IN 47907

Abstract

This paper presents a geometric approach to multichannel, multidimensional sinusoidal identification. A convergence result involving a collection of spectral matrices is given, which yields a method for identifying the signal wavenumbers and amplitude matrices in an arbitrary unknown, possibly nonstationary, noise environment. These results generalize and provide a geometric interpretation of some point spectrum ideas of Geronimus and Capon.

1. Introduction

In 1948 Geronimus [6],[p.216,7] discovered that the inverse of the sum of the (modulus squared) orthogonal polynomials corresponding to a Toeplitz matrix converges to the point spectrum as the matrix order goes to infinity. In 1969 Capon [2] independently generalized part of Geronimus' result to the random field setting, as a fixed order point spectrum estimator. In signal processing terminology, this Geronimus-Capon result is most commonly known as the maximum likelihood or minimum variance spectral estimate. For a nice review of Capon's spectral estimator and its applications in signal processing see [11],[12]. A geometric interpretation of the Geronimus-Capon result is given in [4] and generalized to the multichannel setting in [5]. By studying the free energy in the Gaussian model on the semi-axis, physicists [1],[13],[15],[16] have arrived at the problem of computing the sum of the entries of the inverse of a Toeplitz matrix. This represents a special case of the Capon maximum likelihood estimate. In signal processing terminology they are estimating the inverse of the power of a harmonic signal at frequency equal to zero. Their studies have included multichannel and multidimensional settings. In this paper we will present a geometric framework to generalize the Geronimus-Capon result to the multichannel, multidimensional random field setting and we will allow the noise to be nonstationary. In the case of stationary noise, our framework is also intended to present some further insight into this spectral estimator. Our approach is purely geometric and relies only on some elementary facts concerning Hilbert spaces. We will not use any measure-theoretic arguments.

2. Preliminaries

To begin, let $\mathscr{G}$ denote the group consisting of the integer-valued coordinates on $\mathbb{R}^d$, let $\{\sigma_i\}$ be an ordering on $\mathscr{G}$, and let $\mathscr{H} = \mathbb{C}^p$. Consider the p-channel, d-dimensional observation random field

$$Y(\sigma) = X(\sigma) + N(\sigma)$$

where $N(\sigma)$ is a purely nondeterministic, possibly nonstationary random noise field, and $X(\sigma)$ is uncorrelated with $N(\sigma)$, and is a random signal field of the form

$$X(\sigma) = \sum_{m=1}^{k} A_m \, [e^{-i[(\gamma_m,\sigma) + \xi_{1m}]} ,\dots ,e^{-i[(\gamma_m,\sigma) + \xi_{pm}]}]^{tr} .$$

Here $(\gamma,\sigma) = \sigma^{tr}\gamma$ is the usual inner product on $\mathbb{C}^d$, and tr denotes the transpose. (Throughout, $(\cdot,\cdot)$ denotes the inner product on the appropriate Hilbert space.) The $\{A_m\}_1^k$ are nonzero pxp matrices with complex entries, the $\{\gamma_m\}_1^k$ are distinct wavenumbers in $\mathbb{C}^d$, and the $\{\xi_{jm} : 1 \le j \le p \text{ and } 1 \le m \le k\}$ are mutually independent random variables uniformly distributed over $[0, 2\pi]$. Thus $X(\sigma)$ is a homogeneous, deterministic random field. In this paper we will present a geometric framework to generalize the Geronimus-Capon result to identify the signal wavenumbers $\{\gamma_m\}_1^k$ and amplitude matrices $\{A_m\}_1^k$ in an arbitrary unknown possibly nonstationary noise environment.

The correlation function, $R_X(\sigma)$, for $X(\sigma)$ is given by

$$R_X(\alpha,\beta) = E[X(\alpha)X(\beta)^*] = \sum_{m=1}^{k} A_m A_m^* e^{-i(\gamma_m,\alpha-\beta)} = R_X(\alpha-\beta) \qquad (\alpha,\beta \in \mathscr{G}) \quad (2.1)$$

where E is the expectation. Let $R(A_m^*)$ denote the range space of A_m^* and define the space $\mathscr{H}_X$ by

$$\mathscr{H}_X \overset{\Delta}{=} \bigoplus_{m=1}^{k} R(A_m^*) \, .$$

On this space we define the unitary representation $W(\sigma)$ of the group $\mathscr{G}$ by

$$W(\sigma) \overset{\Delta}{=} \text{diagonal } [e^{i(\gamma_1,\sigma)} I_1 \,,\dots,\, e^{i(\gamma_k,\sigma)} I_k]$$

where I_m is the identity on $R(A_m^*)$. Let C_X be the operator from $\mathscr{H}_X$ to $\mathscr{H}$ defined by $C_X \overset{\Delta}{=} [A_1 ,\dots, A_k]$. Then (2.1) may be expressed as

$$R_X(\alpha,\beta) = R_X(\alpha-\beta) = C_X \, W(\alpha)^* W(\beta) \, C_X^* = C_X \, W(\alpha-\beta)^* C_X^* \quad . \qquad (2.2)$$

Notice that $W(\sigma)$ has k distinct eigenvalues $\{e^{i(\gamma_m,\sigma)}\}_1^k$, each with multiplicity equal to the dimension of the corresponding range space.

Throughout, we assume that the noise covariance $R_N(\sigma_i,\sigma_j) = E[N(\sigma_i),N(\sigma_j)^*]$ satisfies the following condition:

$$\delta\sum_i ||u(\sigma_i)||^2 \le \sum_{i,j}(R_N(\sigma_i,\sigma_j)u(\sigma_j),u(\sigma_i)) \le M\sum_i ||u(\sigma_i)||^2 \tag{2.3}$$

where $\delta > 0$ and $\{u(\sigma_i) : \sigma_i \epsilon \mathcal{G}\}$ is any sequence with values in $\mathcal{H}$ with compact support. Let $\ell^2(\mathcal{H})$ be the set of all square summable sequences $u = \{u(\sigma) : \sigma\epsilon\mathcal{G}\}$ with values in $\mathcal{H}$. Let $\hat{R}$ on $\ell^2(\mathcal{H})$ be the positive operator defined by

$$\hat{R}u = (\hat{R}u)(\sigma) = \sum_{\sigma_j \in \mathcal{G}} R_N(\sigma,\sigma_j)u(\sigma_j) \ .$$

Condition (2.3) implies that $\hat{R}$ is a bounded, invertible positive operator. Let Φ on $\ell^2(\mathcal{H})$ be the positive square root of $\hat{R}$. Let $S(\gamma)$ on $\ell^2(\mathcal{H})$ be the unitary representation of the group $\mathcal{G}$ defined by

$$S(\gamma)u = (S(\gamma)u)(\sigma) = S(\gamma)\ \{u(\sigma) : \sigma \in \mathcal{G}\} = \{u(\sigma-\gamma) : \sigma \in \mathcal{G}\} = u(\sigma-\gamma)$$

where u is in $\ell^2(\mathcal{H})$ and γ is in $\mathcal{G}$. Obviously, $S(\gamma)$ is unitary for all γ, and $S(\sigma_i + \sigma_j) = S(\sigma_i)S(\sigma_j)$ for all σ_i,σ_j in $\mathcal{G}$. Let C_N from $\ell^2(\mathcal{H})$ to $\mathcal{H}$ and $T(\gamma)$ on $\ell^2(\mathcal{H})$ be the operators defined by

$$C_N^* h = \Phi\delta(\sigma)h \qquad \text{and} \qquad T(\gamma) = \Phi S(\gamma)\Phi^{-1}$$

where h is in $\mathcal{H}$ and $\delta(\sigma)$ is the Kronecker delta function. Clearly, $T(\gamma)$ is a representation of the group $\mathcal{G}$; that is, $T(0) = I$ and $T(\sigma_i + \sigma_j) = T(\sigma_i)\,T(\sigma_j)$ for all σ_i,σ_j in $\mathcal{G}$. Finally, we note that if N is wide sense stationary (wss); that is, $R_N(\sigma_i,\sigma_j) = R_N(\sigma_i-\sigma_j)$, then $S(\gamma)\hat{R} = \hat{R}S(\gamma)$. Since the square root Φ can be obtained as a limit of polynomials in $\hat{R}$, this implies that $S(\gamma)$ also commutes with Φ for all γ in $\mathcal{G}$. Therefore, if N is wss, then $T(\gamma) = S(\gamma)$ is a unitary operator for all γ in $\mathcal{G}$.

We claim that

$$R_N(\alpha,\beta) = C_N T(\alpha)^* T(\beta) C_N^* \qquad (\text{ for all } \ \alpha,\beta \in \mathcal{G}) \ . \tag{2.4}$$

To prove (2.4) let f and g be elements of $\mathcal{H}$. Then (2.4) follows from

$$(C_N T(\alpha)^* T(\beta) C_N^* f,g) = (\Phi^2 S(\beta)\delta(\sigma)f, S(\alpha)\delta(\sigma)g) = (\hat{R}\delta(\sigma-\beta)f, \delta(\sigma-\alpha)g) =$$
$$\sum_{i,j}(R_N(\sigma_i,\sigma_j)\delta(\sigma_j-\beta)f, \delta(\sigma_i-\alpha)f) = (R_N(\alpha,\beta)f,g) \ .$$

Let $\mathcal{F}$ be the quarter plane subset of $\mathcal{G}$ defined by $\mathcal{F} = \{\sigma = (\sigma_1, \ldots, \sigma_d) \in \mathcal{G} : \sigma_1 \ge 0, \ldots, \sigma_d \ge 0\}$. Let $\ell_+(\mathcal{H})$ be the subspace of $\ell^2(\mathcal{H})$ defined by $\ell_+(\mathcal{H}) = \{u(\sigma) \in \ell^2(\mathcal{H}) : \sigma \in \mathcal{F}\}$. Finally, let $\mathcal{M} = \Phi\ell_+(\mathcal{H})$. Since Φ is invertible, $\mathcal{M}$ is closed. We claim that

$$\text{strong} \lim_{n\to\infty} P_{\mathcal{M}} T(\gamma)^{*n} = 0 \quad (\text{for all } \gamma \neq 0 \text{ in } \mathcal{F}) \tag{2.5}$$

where $P_{\mathcal{M}}$ is the orthogonal projection onto $\mathcal{M}$. To see this, notice that

$$P_{\mathcal{M}} = \Phi \hat{R}_+^{-1} P_+ \Phi$$

where $\hat{R}_+$ is the positive invertible operator on $\ell_+(\mathcal{H})$ defined by $\hat{R}_+ = P_+ \hat{R} \mid \ell_+(\mathcal{H})$ and P_+ is the orthogonal projection onto $\ell_+(\mathcal{H})$. Now, equation (2.5) follows from $T(\gamma)^* = \Phi^{-1} S(\gamma)^* \Phi$ and the fact that the bilateral shift $S(\gamma)$ satisfies (see [3],[14])

$$\text{strong} \lim_{n\to\infty} P_+ S(\gamma)^{*n} = 0 \quad (\text{for all } \gamma \neq 0 \text{ in } \mathcal{F}) \ .$$

From (2.2) and (2.4) the observation correlation function can be expressed as

$$R_Y(\sigma_i,\sigma_j) = R_X(\sigma_i - \sigma_j) + R_N(\sigma_i,\sigma_j) = C\, V(\sigma_i)^* V(\sigma_j)\, C^* \quad (\sigma_i,\sigma_j \in \mathcal{G}) \tag{2.6}$$

where

$$V(\sigma) = W(\sigma) \oplus T(\sigma) \text{ on } \mathcal{K} \overset{\Delta}{=} \mathcal{K}_X \oplus \mathcal{K}_N = \bigoplus_{m=1}^{k} R(A_m^*) \oplus \ell^2(\mathcal{H}) \quad , \tag{2.7}$$

is a representation of the group $\mathcal{G}$, and $C \overset{\Delta}{=} [C_X, C_N]$ maps $\mathcal{K}$ into $\mathcal{H}$. In the case of wide sense stationary noise $T(\sigma) = S(\sigma)$ on $\ell^2(\mathcal{H})$. In the stationary setting the representation $V(\sigma)$ is usually referred to as the *Naimark dilation* of $R_Y(\sigma_i,\sigma_j)$; see [3],[14] for further details. The decomposition of $V(\sigma)$ into the direct sum of a unitary operator $W(\sigma)$ and a shift $S(\sigma)$ is known as the *Wold decomposition* [3],[10],[14].

3. The Main Result

In this section we present a convergence property which will identify the signal wavenumbers $\{\gamma_m\}_1^k$ and squared amplitudes $\{A_m A_m^*\}_1^k$. If $B_1, B_2, \ldots, B_n$ is a set of matrices (or vectors), then $[B_1, B_2, \ldots, B_n]^{tr}$ denotes the block column matrix whose j^{th} column is the matrix (or vector) B_j. Recall that $\bigvee$ is the closed linear span. We begin with the following lemma, which is a classical application of the projection theorem.

Lemma 1. *Let* $\{C_i\}_1^n$ *be any collection of operators from a Hilbert space* $\mathcal{K}$ *into a Hilbert space* $\mathcal{H}$, *and let* $\mathcal{K}_n$ *be the subspace of* $\mathcal{K}$ *defined by* $\mathcal{K}_n \overset{\Delta}{=} \bigvee_1^n C_i^* \mathcal{H}$. *Let* $\hat{R}_n$ *be the nxn block grammian matrix with* $(i,j)^{th}$ *element* $C_i C_j^*$, *and assume that* $\hat{R}_n$ *is invertible. Then for any* $y \in \mathcal{K}$, *we have*

$$P_n y = [C_1^*, \ldots, C_n^*]\, \hat{R}_n^{-1}\, [C_1 y, \ldots, C_n y]^{tr} \quad , \tag{3.1}$$

where P_n *is the orthogonal projection onto* $\mathcal{K}_n$. *Moreover,*

$$(P_n y, y) = (\hat{R}_n^{-1}\,[C_1 y, \dots, C_n y]^{tr}, [C_1 y, \dots, C_n y]^{tr}) . \tag{3.2}$$

Throughout, $\hat{R}_n = \{R_Y(\sigma_i, \sigma_j)\}_{i,j=1}^n$ is the nxn signal-plus-noise correlation block matrix with pxp matrix-valued elements determined by (2.6), where $\{\sigma_j\}_1^\infty$ is a distinct set in $\mathcal{G}$. Notice that the $(i,j)^{th}$ entry of $\hat{R}_n$ is $\hat{R}_n(i,j) = R_Y(\sigma_i, \sigma_j) = R_Y(\sigma_j, \sigma_i)^*$. Furthermore, the strict positivity of $R_N(\sigma_i, \sigma_j)$ (see (2.3)) implies that the nxn block matrix $\hat{R}_N$ with entries $\{R_N(\sigma_i, \sigma_j) : 1 \leq i,j \leq n\}$ is invertible. Since $R_Y = R_X + R_N$ this implies that $\hat{R}_n$ is invertible. At this point we require some further notation. For any $\alpha = (\alpha_1, \dots, \alpha_d) \in \mathcal{G}$ and $z = [\lambda_1, \dots, \lambda_d]^{tr}$ in $\mathbb{C}^d$ denote $z^\alpha = \prod_{j=1}^{d} \lambda_j^{\alpha_j}$. Throughout this paper we always assume that z is a vector in $\mathbb{C}^d$ whose components all have modulus one; that is, $|\lambda_j| = 1$ for all j=1,2,...d. For any distinct sequence of vectors $\{\sigma_j = (\sigma_{j1}, \dots, \sigma_{jd})\}_{j=1}^n$ in $\mathcal{G}$, let $M_n(z)$ be the operator on $\mathcal{H}$ given by

$$M_n(z) \overset{\Delta}{=} [z^{\sigma_1}I, \dots z^{\sigma_n}I]\,\hat{R}_n^{-1}\,[\bar{z}^{\sigma_1}I, \dots, \bar{z}^{\sigma_n}I]^{tr}$$

where I is the identity on $\mathcal{H}$. Obviously, $M_n(z)$ is invertible for all such z. Finally, for any r^{th} signal component with amplitude matrix A_r and wavenumber $\gamma_r = (\gamma_{r1}, \dots, \gamma_{rd})$ denote $z_r = [e^{i\gamma_{r1}}, \dots, e^{i\gamma_{rd}}]^{tr}$. Our aim is to show that the operator $(M_n(z)h, h)$ for h in $\mathcal{H}$ is bounded as n approaches infinity if and only if $z = z_r$ and h is in $R(A_r)$ for some r=1,2,...,k. Our first step in this direction is the following lemma.

Lemma 2. *For any subset* $\{\sigma_j\}_1^\infty$ *of* $\mathcal{G}$ *the sequence* $(M_n(z)h, h)$ *is increasing for all* z *and for all* h *in* $\mathcal{H}$.

Proof. There are a number of ways to prove this lemma. Our proof will contain results to be used in the proof of the next theorem. To begin, let $C_i = CV(\sigma_i)^*$ for all i, and as in Lemma 1, let $\mathcal{H}_n$ be the subspace of $\mathcal{H}$ defined by $\mathcal{H}_n \overset{\Delta}{=} \bigvee_1^n C_i^* \mathcal{H}$. We define the vector $\phi_n(z,h)$ in $\mathcal{H}_n$ by

$$\phi_n(z,h) \overset{\Delta}{=} [V(\sigma_1)C^*, \dots, V(\sigma_n)C^*]\,\hat{R}_n^{-1}\,[\bar{z}^{\sigma_1} I, \dots, \bar{z}^{\sigma_n}I]^{tr}h \quad . \tag{3.3}$$

Set $\phi_n = \phi_n(z,h)$. Operating on ϕ_n with $C_j = CV(\sigma_j)^*$ for $j \leq n$, and recalling (2.6) gives

$$C_j\phi_n = CV(\sigma_j)^*\,[V(\sigma_1)C^*, \dots, V(\sigma_n)C^*]\,\hat{R}_n^{-1}\,[\bar{z}^{\sigma_1}I, \dots, \bar{z}^{\sigma_n}I]^{tr}h =$$

$$[R_Y(\sigma_j, \sigma_1), \dots, R_Y(\sigma_j, \sigma_n)]\,\hat{R}_n^{-1}\,[\bar{z}^{\sigma_1} I, \dots, \bar{z}^{\sigma_n}]^{tr}\,h = \bar{z}^{\sigma_j}h . \tag{3.4}$$

The last equality in (3.4) follows from the fact that the j^{th} row of $\hat{R}_n$ is

$$[R_Y(\sigma_j, \sigma_1), \dots, R_Y(\sigma_j, \sigma_n)] = [0, \dots, 0, I, 0, \dots, 0]\,\hat{R}_n .$$

From (3.3) and (3.4) it follows that $(\phi_n, \phi_n) = (M_n(z)h, h)$ for all n. Lemma 1 along with (3.4) shows that $P_j\phi_n = \phi_j$. To see this write

$$P_j \ \phi_n = [C_1^*, \dots, C_j^*]\ \hat{R}_j^{-1}\ [C_1\phi_n, \dots, C_j\phi_n]^{tr} =$$

$$[V(\sigma_1)C^*, \dots, V(\sigma_j)C^*]\ \hat{R}_j^{-1}\ [\bar{z}^{\sigma_1} I, \dots, \bar{z}^{\sigma_j} I]^{tr}\ h = \phi_j \ . \tag{3.5}$$

So using $(M_j(z)h,h) = (\phi_j,\phi_j)$ we obtain

$$(M_j(z)h,h) = ||\phi_j(z,h)||^2 \leq ||\phi_n(z,h)||^2 = (M_n(z)h,h) \qquad (j \leq n) \ . \tag{3.6}$$

This completes the proof.

We now state the main result of this work.

Theorem. *Let* $\mathcal{F} = \{(\alpha_1, \dots, \alpha_d) \in \mathcal{G} : \alpha_i \geq 0$ *for all* $i\}$ *be a quarter plane,* $\{\sigma_j\}_1^\infty$ *be any ordering of* $\mathcal{F}$, *and let* z *be any vector in* $\mathbb{C}^d$ *whose components all have modulus one. Then* $\lim_{n\to\infty} (M_n(z)h,h) < \infty$ *if and only if* $h \in R(A_r)$ *and* $z = z_r$ *for some* $r=1,2,\dots,k$. *In this case,*

$$\lim_{n\to\infty} A_r^* M_n(z) A_r = Q_r \ , \tag{3.7}$$

where Q_r *is the orthogonal projection onto* $\mathcal{H} \ominus \ker(A_r) = R(A_r^*)$.

Proof. Let $\mathcal{K}_\infty = \bigvee\{V(\sigma_j)C^*\mathcal{H} : \sigma_j \in \mathcal{F}\}$. Suppose first that $y \in \mathcal{K}$ has the form $y = f \oplus 0$ where f is in $R(A_r^*)$, and let $\{C_i\}_1^n$ and $\mathcal{K}_n$ be defined as in the proof of Lemma 2. Then $C_i y = C_X W(\sigma_i)^* y = \bar{z}_r^{\sigma_i} A_r f$. Lemma 1 and (3.3) imply that $P_n f = \phi_n(z_r, A_r f)$. By substituting $z = z_r$ and $h = A_r f$ into the last equation in (3.6) we obtain $(P_n f,f) = (M_n(z_r)A_r f, A_r f)$. So if P_∞ is the orthogonal projection onto $\mathcal{K}_\infty = \bigvee\{\mathcal{K}_n : n \geq 0\}$, then

$$(P_\infty f,f) = \lim_{n\to\infty} (A_r^* M_n(z_r) A_r f,f) \ . \tag{3.8}$$

Using $h = A_r f$ we have $\lim_{n\to\infty} (M_n(z_r)h,h) < \infty$ for h in $R(A_r)$.

Consider now the case where $\lim_{n\to\infty} (M_n(z)h,h) < \infty$ for some h in $\mathcal{H}$ and $z = [\lambda_1, \dots, \lambda_d]^{tr}$ with $|\lambda_j| = 1$ for all j. Equation (3.6) and $P_j\phi_n = \phi_j$ for $j \leq n$, where $\phi_n = \phi_n(z,h)$, imply that

$$||\phi_n - \phi_j||^2 = ||\phi_n - P_j\phi_n||^2 = ||\phi_n||^2 - ||P_j\ \phi_n||^2 =$$

$$||\phi_n||^2 - ||\phi_j||^2 = (M_n h,h) - (M_j h,h) \ .$$

The fact that $(M_n(z)h,h)$ is increasing and bounded above implies $\{\phi_n\}$ is Cauchy. Thus, $\phi_n \to \phi$ as $n \to \infty$, where ϕ is in $\mathcal{K}_\infty$. To prove the desired result, let e_q be the standard basis element in $\mathcal{G}$ of the form $e_q = (0, \dots, 0,1,0, \dots, 0)$ where the q^{th}

component is one. We claim that $\phi = \phi(z,h)$ is in $\mathscr{H}_X$, and ϕ is an eigenvector for $W(e_q)^*$, regardless of the value of q. To see this, notice that for all large $n > m$ and g in $\mathscr{H}$, equation (3.4) for $\sigma_j = \sigma_m + e_q$ gives

$$(V(e_q)^* \phi_n, V(\sigma_m)C^* g) = (CV(\sigma_m + e_q)^* \phi_n, g) = (\overline{z}^{\sigma_m + e_q} h, g) = \overline{\lambda}_q \overline{z}^{\sigma_m} (h,g) \tag{3.9}$$

Similarly, write

$$(\overline{\lambda}_q \phi_n, V(\sigma_m)C^* g) = \overline{\lambda}_q (CV(\sigma_m)^* \phi_n, g) = \overline{\lambda}_q \overline{z}^{\sigma_m}(h,g) \ . \tag{3.10}$$

Since ϕ_n converges strongly to ϕ, equations (3.9) and (3.10) imply that $V(e_q)^* \phi - \overline{\lambda}_q \phi$ is orthogonal to $V(\sigma_m)C^* \mathscr{H}$ for all m. Because the closed linear span of all $V(\sigma_m)C^* \mathscr{H}$ equals $\mathscr{H}_\infty$ we have

$$P_\infty V(e_q)^* \phi = \overline{\lambda}_q \phi \qquad \text{(for all q=1,2,...,d)} . \tag{3.11}$$

From the fact that $\mathscr{H}_\infty$ is invariant for $V(e_q)$ it follows that $P_\infty V(e_q)^* = P_\infty V(e_q)^* P_\infty$. Using this in (3.11) we obtain

$$P_\infty V(e_q)^{*n} \phi = \overline{\lambda}_q^n \phi \qquad \text{(for all q and } n \geq 0) \ .$$

Now let $\phi = \phi_1 \oplus \phi_2$, where ϕ_1 is in $\mathscr{H}_X$ and ϕ_2 is in $\mathscr{H}_N$. Equations (2.5), (2.7) and the fact that $\mathscr{H}_\infty \subseteq \mathscr{H}_X \oplus \mathscr{M}$ imply that

$$\begin{gathered} ||\phi_1||^2 + ||\phi_2||^2 = ||\overline{\lambda}^n \phi||^2 = ||P_\infty V(e_q)^{*n} \phi||^2 \leq \\ ||\phi_1||^2 + ||P_{\mathscr{M}} T(e_q)^{*n} \phi_2||^2 \rightarrow ||\phi_1||^2 \ . \end{gathered}$$

Thus, $\phi = \phi_1$ is in $\mathscr{H}_X$. Equation (3.11) implies that $||\phi_1|| = ||P_\infty W(e_q)^* \phi_1|| \leq ||\phi_1||$, so we have equality. This gives $P_\infty W(e_q)^* \phi_1 = W(e_q)^* \phi_1$. Hence, $W(e_q) = \overline{\lambda}_q \phi$ for all q. Because the signal wavenumbers $\{\gamma_m\}_1^k$ are all distinct, and ϕ is an eigenvector for $W(e_q)$ with eigenvalue λ_q for all q, it follows that $\phi = \phi(z,h)$ is in $R(A_r^*)$ and $z = z_r$ for some r=1,2,...,k. Finally, using (3.4) for $n > j$ we have $\overline{z}^{\sigma_j} A_r \phi = C_j \phi = C_j \phi_n = \overline{z}^{\sigma_j} h$. Hence h is in $R(A_r)$. In fact, this analysis shows that ϕ_n converges to the unique element ϕ in $R(A_r^*)$ satisfying $h = A_r \phi$. In conclusion, we have thus far shown that $\lim_{n \rightarrow \infty} (M_n(z)h,h)$ is finite if and only if h is in $R(A_r)$ and $z = z_r$ for some r=1,2,...,k.

To obtain (3.7) first let $h = A_r f$ for some f in $R(A_r^*)$ and let $z = z_r$. Using $P_n f = \phi_n(z_r, A_r f)$ we obtain $P_\infty f = \phi(z_r, A_r f)$. By our previous discussion this implies that $P_\infty f$ is in $R(A_r^*)$. Thus, P_∞ maps $R(A_r^*)$ into $R(A_r^*)$. From (3.8) and the fact that $M_n(z)$ is positive, nonsingular and increasing, it follows that $P_\infty f = 0$ if and only if $f = 0$. Therefore P_∞ maps $R(A_r^*)$ onto itself. In particular, $R(A_r^*) \subseteq \mathscr{H}_\infty$, so that $P_\infty f = f$ for $f \in R(A_r^*)$. (From this it also follows that $\mathscr{H}_X \oplus \mathscr{M} = \mathscr{H}_\infty$.) Finally, (3.7) follows from (3.8). This completes the proof.

The following corollary represents our extension of the Geronimus-Capon result to the multichannel, multidimensional setting.

Corollary. *Let z be any vector in* $\mathbb{C}^d$ *whose components are all of modulus one. Then*

$$\lim_{n\to\infty} M_n(z)^{-1} = \sum_1^k A_m A_m^* \,\delta(z - z_r) \tag{3.12}$$

where $\delta(x) = I$ *if* $x = 0$ *and* $\delta(x) = 0$ *otherwise.*

Proof. We will only sketch a proof of this corollary, since the ideas are similar to those in [5]. Let $z = z_r$ and let $\lambda_{1n} \leq \lambda_{2n} \leq \cdots \leq \lambda_{pn}$ be the eigenvalues of $M_n(z)$. Let j be the rank of A_r and E_n be the orthogonal projection onto the eigenvector space corresponding to the eigenvalues $\{\lambda_{in} : i = 1,\dots,j\}$. By consulting Lemma 2 of [5] or using the minimax definition of eigenvalues along with the fact that $M_n(z)$ is increasing, it follows that E_n converges to the orthogonal projection onto $R(A_r)$. Moreover, $\{\lambda_{in}\}_{n=1}^{\infty}$ is an increasing sequence for each i ($1 \leq i \leq p$), and as $n\to\infty$ the eigenvalues λ_{in} converge for $i \leq j$ and diverge for $i > j$. Thus,

$$M_n(z)^{-1} = E_n M_n(z)^{-1} E_n + (I - E_n) M_n(z)^{-1} (I - E_n) \tag{3.13}$$

converges to a positive operator M_∞^{-1} whose range equals $R(A_r)$. (The last term in (3.13) converges to zero.) So by performing the appropriate pseudoinverses on A_r and A_r^* in (3.7) we obtain (3.12). Clearly, $M_n(z)^{-1}$ converges to zero if $z \neq z_r$.

4. Discussion

The above theorem provides a method for identifying the signal wavenumbers and squared amplitude matrices corresponding to a harmonic signal in an arbitrary nonstationary noise field. Specifically, let $z = [e^{i\omega_1},\dots,e^{i\omega_d}]^{tr}$, and as before let $\lambda_{1n}(z) \leq \lambda_{2n}(z) \leq \cdots \leq \lambda_{pn}(z)$ be the eigenvalues of $M_n(z)$. Then by our previous discussion $\omega \overset{\Delta}{=} (\omega_1, \dots, \omega_d)$ is a signal wavenumber (that is, $\omega = \gamma_r$ for some r) if and only if $\lim_{n\to\infty} \lambda_{1n}(z) < \infty$. (Recall that $\lambda_{1n}(z)$ is an increasing sequence.) The rank of the corresponding amplitude matrix A_r is equal to the largest integer j such that $\lim_{n\to\infty} \lambda_{jn}(z) < \infty$, and $A_r A_r^*$ is approximately equal to $E_n M_n(z)^{-1} E_n$ for large n. Therefore, one determines the set of signal wavenumbers by letting ω run through the d-dimensional torus formed by $[0,2\pi]$, and determining the values of z for which the sequence $\lambda_{1n}(z)$ converges. At these points of convergence we can then estimate the corresponding amplitude matrices $A_r A_r^*$ by the appropriate restricted inverse of $M_n(z)$.

The original intent of Capon [2] was to develop a spectral estimator which had better resolving properties than the straightforward periodogram approach. It is easy to see why the Geronimus-Capon spectral estimate will perform better than the straightforward discrete Fourier transform (DFT) of $R_Y(\sigma)$ in the single channel, 1-dimensional setting for wide sense stationary noise that is white [2],[11],[12]. Now, let us present a geometric explanation of why this will also be the case in the multichannel, multidimensional colored noise setting. To this end, assume that $N(\sigma)$ is wss, so that $R_N(\sigma_i,\sigma_j) = R_N(\sigma_i - \sigma_j)$ for all σ_i,σ_j in $\mathcal{G}$. Recall that in this case $T(\sigma) = S(\sigma)$ for all σ in $\mathcal{G}$, and $V(\sigma) = W(\sigma) \oplus T(\sigma)$ is a unitary representation of the group $\mathcal{G}$.

So, if $R_Y(\alpha,\beta) = R_Y(\alpha-\beta)$, then equations (2.3), (2.4) and (2.7) imply that

$$R_y(\sigma) = CV(\sigma)^* C^* \qquad (\text{for all } \sigma \in \mathscr{G}) \quad . \tag{4.1}$$

Let us define the contraction V_n on $\mathscr{H}$ by

$$V_n(z) = (1/n^d) \sum_{\sigma \in \mathscr{F}_n} V(\sigma)^* z^\sigma \ , \tag{4.2}$$

where $\mathscr{F}_n \overset{\Delta}{=} \{(\alpha_1, \ldots, \alpha_d) \in \mathscr{G} : 0 \leq \alpha_i < n \text{ for all } i\}$, and $z = [e^{i\omega_1}, \ldots, e^{i\omega_d}]^{tr}$. Then, generalizing the ergodic theorem in Halmos [8] (due to F. Riesz) to arbitrary unitary representations of the group $\mathscr{G}$, it follows that V_n converges strongly to the orthogonal projection onto the space $\{x : V(e_1)x = e^{i\omega_1}x, \cdots, V(e_d)x = e^{i\omega_d}x\}$. Thus, by the Wold decomposition of $V(\sigma) = W(\sigma) \oplus S(\sigma)$ and the above ergodic theorem

$$\text{strong} \lim_{n\to\infty} V_n(z) = \sum_{r=1}^{k} \delta(z - z_r)\hat{Q}_r \tag{4.3}$$

where $\hat{Q}_r$ is the orthogonal projection onto $R(A_r^*)$. (Recall that $S(e_q)$ has no eigenvalues; see [14].) Multiplying (4.2) on the left by C and on the right by C^*, along with (4.1) gives

$$\lim_{n\to\infty} (1/n^d) \sum_{\sigma \in \mathscr{F}_n} R_Y(\sigma)\bar{z}^\sigma = \sum_{r=1}^{k} A_r A_r^* \delta(z - z_r) \ . \tag{4.4}$$

Equation (4.4) is the limit of the standard DFT spectral estimates, and is the same as (3.12) obtained by the Geronimus-Capon estimate. Since $M_n(z)^{-1} \geq M_{n+1}(z)^{-1}$ the sequence $M_n(z)^{-1}$ converges monotonically to the point spectrum. Therefore, $M_n(z)^{-1}$ always provides an upper bound for the point spectrum, and the points of convergence are more readily identified. The DFT approach has the advantage that no inversions are required. The Geronimus-Capon estimate has the advantage that it utilizes the projection theorem to yield the best possible estimate based on the data. Hence it is natural to expect that it will perform better than the DFT method for point spectrum estimation.

Acknowledgement This work was supported in part by the Office of Naval Research Grant Number N00014-89-J-1747.

References

[1] Adamjan, V.M. "Asymptotic properties for positive and Toeplitz matrices and operators", *Operator Theory: Advances and Applications*, **43**, Birkhauser Verlag, Basel, 17-38, 1990.

[2] Capon, J. "High-resolution frequency-wavenumber spectrum analysis", *Proceedings of the IEEE*, **57**, 1408-1418, 1969.

[3] Foias, C. and Frazho, A.E. *The Commutant Lifting Approach to Interpolation Problems* , Operator Theory: Advances and Applications Vol. 44, Birkhauser Verlag, Basel, 1990.

[4] Foias, C., Frazho, A.E. and Sherman, P.J. "A geometric approach to the maximum likelihood spectral estimator for sinusoids in noise", *IEEE Transactions on Information Theory,* **IT-34**(5), 1066-1070, September, 1988.

[5] Foias, C., Frazho, A.E. and Sherman, P.J. "A new approach for determining the spectral data of multichannel harmonic signals in noise", *Mathematics of Control, Signals, and Systems,* **3**, 31-43, January, 1990.

[6] Geronimus, L.Y. "Polynomials orthogonal on a circle and their application", *Zapiski Nauchno-issledovatel'skogo in-ta matematiki i mekhaniki i KhMO,* **19**, 35, 1948.

[7] Geronimus, L.Y. *Orthogonal Polynomials*, Consultants Bureau, New York, 1961.

[8] Halmos, P.R. *Finite Dimensional Vector Spaces*, Van Nostrand, Princeton, 1958.

[9] Helson, H. and Lowdenslager, D. "Prediction theory and Fourier series in several variables", *Acta Math.*, **99**, 165-202,1958.

[10] Helson, H. and Lowdenslager, D. "Prediction theory and Fourier series in several variables,II", *Acta Math.*, **106**, 175-213,1961.

[11] Lacoss, R.T. "Data adaptive spectral analysis methods", *Geophysics,***36**, 661-675, 1971.

[12] Marple, L.S.,Jr. *Digital Spectral Analysis with Applications*, Prentice Hall, 1987.

[13] Sanovich, A.L. and Spitkovskii, I.M. "On block Toeplitz matrices and properties of the Gauss model on the half-line related to them", (Russian), *Teoret. Mat. Fiz.*, **63**, 154-160, 1985.

[14] Sz.-Nagy, B. and Foias, C. *Harmonic Analysis of Operators on Hilbert Space*, North Holland, Amsterdam, 1970.

[15] Vladimirov, V.S. and Volovich, I.V. "The diophantic moment problem, orthogonal polynomials, and some models of statistical physics", in *Lecture Notes in Mathematics*, Springer Verlag, Vol. 1043, 289-292.

[16] Vladimirov, V.S. and Volovich, I.V. "On the model of statistical physics", (Russian), *Teoret. Mat. Fiz.*,**54**(1),8-22,1983.

ROBUST STABILITY OF TWO-DIMENSIONAL DISCRETE INTERVAL MATRICES

P. Bauer
Laboratory of Image and Signal Analysis, Department of Electrical Engineering
University of Notre Dame, Notre Dame, Indiana 46556

Abstract:

In this paper robust BIBO-stability of 2-D discrete interval systems is analyzed. The test for one particular shift-invariant / shift-variant 2-D Roesser interval matrix involves one 2-D linear shift-invariant stability test only. The stability condition derived also provides insight into robustness and margin of stability of the shift-invariant/shift-variant system and it can readily be extended to the general m-D case.

The results of this paper generalizes an implication of the Perron- Frobenius Theorem [1-3] from 1-D shift-invariant systems to shift-variant and / or multi-dimensional systems.

I. Introduction

In the last two decades a vast amount of research was devoted to the area of multi-dimensional systems. The rapid increase of the applicability of systems in areas such as image processing, tomography, 3-D radar, seismographic data processing, lumped networks etc. demonstrate the significance of this field of research. Most of these research activities were concentrated on linear shift-invariant discrete and continuous systems. An overview concerning the stability analysis of linear m-D systems is presented in a comprehensive study [4].

The subject of robust stability of interval systems has also attracted considerable attention in recent years. Comprehensive overviews of this subject can be found in [1,5].

Unfortunately the stability analysis of shift-varying or multi-dimensional systems has not been investigated thoroughly. The applications for shift-varying and /or multi-dimensional recursive systems are in adaptive filters and systems, the convergence of adaptive and recursive algorithms, even in robust linear and nonlinear recursive systems.

For 1-D discrete shift-variant recursive systems, some stability conditions were derived in [6]. The extension of the 1-D results to robust shift-invariant multi-dimensional systems has been reported in [8]. Results on robust stability of m-D polynomials can be found in [7].

This paper analyzes robust stability of m-D shift-varying or shift-invariant discrete systems represented by the Roesser system matrix. Also the issue of margin of stability of classes of shift-varying systems is addressed. The basic philosophy behind the methods developed in this paper is the construction of a linear shift-invariant system, the states of which constitute an upper bound to the shift-varying system states.

II. The Main Result

In this section, a stability condition for multi-input multi-output 2-D state space systems is introduced. The following notation is used for 2-D state space systems:

A^-, B^-, C^-, D^-: Shift-varying state space system matrix and state matrices

$x^-(n_1, n_2)$: 2-D shift-varying state vector

$x^{-h}(n_1, n_2)$: 2-D shift-varying horizontally propagating state vector

$x^{-v}(n_1, n_2)$: 2-D shift-varying vertically propagating state vector

$y^-(n_1, n_2), u^-(n_1, n_2)$: 2-D shift-varying output vector and input vector.

The discussion in this section will be limited to quarterplane causal shift-variant 2-D systems under zero initial conditions. The superscript of a variable indicates, that the variable describes a shift-variant system. For linear shift-invariant systems, the superscript is omitted. The elements of the matrices A^-, B^-, C^- and D^- are refered to as $a_{ij}^-, b_{ij}^-, c_{ij}^-$ and d_{ij}^-.

The method, which will be developed in this section, does not only guarantee BIBO-stability but also stability of the individual states. This is achieved by proving the existence of an upper bound for the norm of the shift-varying system states, assuming there exists an upper bound for the norm of the input vector . Once stability of the states is established, it is straight forward to show BIBO-stability: Assume M_1 and M_2 are the upper bounds of the norm of the 2-D state vector and the input vector respectively. Then using the 2-D Roesser model [9] for quarter plane discrete recursive digital systems we obtain:

$$y^-(n_1, n_2) = C^- \begin{pmatrix} x^{-h}(n_1, n_2) \\ x^{-v}(n_1, n_2) \end{pmatrix} + D^- u^-(n_1, n_2) \tag{1}$$

with

$$y^-(n_1, n_2) \epsilon \mathcal{R}^g, \quad x^{-h}(n_1, n_2) \epsilon \mathcal{R}^l, \quad x^{-v}(n_1, n_2) \epsilon \mathcal{R}^k, \quad u^-(n_1, n_2) \epsilon \mathcal{R}^f$$
$$C \epsilon \mathcal{R}^{g \times (l+k)}, \quad D \epsilon \mathcal{R}^{g \times f}.$$

From equation (1) we have:

$$\begin{aligned} \| y^-(n_1, n_2) \| &= \| C^- \begin{pmatrix} x^{-h}(n_1, n_2) \\ x^{-v}(n_1, n_2) \end{pmatrix} + D^- u^-(n_1, n_2) \| \\ &\leq \| C^- \| \ \| \begin{pmatrix} x^{-h}(n_1, n_2) \\ x^{-v}(n_1, n_2) \end{pmatrix} \| + \| D^- \| \ \| u^-(n_1, n_2) \| \\ &\leq \| C^- \| M_1 + \| D^- \| M_2 \end{aligned} \tag{2}$$

Hence the output vector is bounded.

Next boundedness of the norm of the states is proven under the assumption of a bounded input and certain conditions for the system matrix elements. Consider the following 2-D shift-varying input-state equation, given by the Roesser model as:

$$\begin{pmatrix} x^{-h}(n_1+1,n_2) \\ x^{-v}(n_1,n_2+1) \end{pmatrix} = A^- \begin{pmatrix} x^{-h}(n_1,n_2) \\ x^{-v}(n_1,n_2) \end{pmatrix} + B^- u^-(n_1,n_2) \tag{3}$$

where

$$A^- \epsilon \mathcal{R}^{(l+k)\times(l+k)}, \quad B^- \epsilon \mathcal{R}^{(l+k)\times f}.$$

Next a 2-D linear shift-invariant state input equation will be defined as:

$$\begin{pmatrix} x^{h}(n_1+1,n_2) \\ x^{v}(n_1,n_2+1) \end{pmatrix} = A \begin{pmatrix} x^{h}(n_1,n_2) \\ x^{v}(n_1,n_2) \end{pmatrix} + Bu(n_1,n_2). \tag{4}$$

where

$$x^h \epsilon \mathcal{R}^l, \quad x^v \epsilon \mathcal{R}^k, \quad u \epsilon \mathcal{R}^f, \quad A \epsilon \mathcal{R}^{(l+k)\times(l+k)}, \quad B \epsilon \mathcal{R}^{(l+k)\times f}.$$

Furthermore assume the matrices A and B of the linear shift-invariant system in (4) to consist of non-negative matrix entries only, i.e. $a_{ij} \geq 0$ (5), $b_{ij} \geq 0$ (6).

Theorem 1:

Consider the set of all shift-variant 2-D systems satisfying equations (1,3) and

$$\mid a_{ij}^- \mid \leq a_{ij}, \quad a_{ij} \geq 0 \tag{7}$$

$$\mid b_{ij}^- \mid \leq b_{ij}, \quad b_{ij} \geq 0. \tag{8}$$

This set of shift-variant systems is internally * and therefore BIBO-stable , iff the linear shift-invariant system in (4-6) is internally (and therefore BIBO stable). Furthermore if the conditions

$$\mid u_i^-(n_1,n_2) \mid \leq u_i(n_1,n_2); \quad i = 1,\ldots,f \tag{9}$$

are also satisfied, then the norm of the states of the linear shift-invariant system in (4) is an upper bound to the norm of the shift-variant system states in (3).

Proof:

At first it will be shown, that from equations (7-9) and

$$\mid x_i^{-h,v}(n_1,n_2) \mid \leq x_i^{h,v}(n_1,n_2) \tag{10}$$

it follows, that

$$\mid x_j^{-h}(n_1+1,n_2) \mid \leq x_j^h(n_1+1,n_2) = \mid x_j^h(n_1+1,n_2) \mid, \quad j = 1,\ldots,l \tag{11}$$

---

* A system is called internally stable, if under a bounded input signal the system states are bounded.

$$| x_j^{-v}(n_1, n_2 + 1) | \leq x_j^{v}(n_1, n_2 + 1) = | x_j^{v}(n_1, n_2 + 1) |, \quad j = 1, \ldots, k \tag{12}$$

where indices i and j denote the i-th or j-th component of the corresponding vectors.

The proof is as follows:

$$\begin{aligned}
& | x_j^{-h}(n_1 + 1, n_2) | = \\
& \leq | a_{j,1}^{-} x_1^{-h}(n_1, n_2) | + | a_{j,2}^{-} x_2^{-h}(n_1, n_2) | + \ldots + | a_{j,l+1}^{-} x_1^{-v}(n_1, n_2) | + \ldots \\
& + | a_{j,l+k}^{-} x_k^{-v}(n_1, n_2) | + | b_{j1}^{-} u_1^{-}(n_1, n_2) | + \ldots + | b_{j,f}^{-} u_f^{-}(n_1, n_2) | \\
& \leq a_{j,1} x_1^{h}(n_1, n_2) + a_{j,2} x_2^{h}(n_1, n_2) + \ldots + a_{j,l+1} x_1^{v}(n_1, n_2) + \ldots \\
& + a_{j,l+k} x_k^{v}(n_1, n_2) + b_{j,1} u_1(n_1, n_2) + \ldots + b_{j,f} u_f(n_1, n_2) \\
& = x_j^{h}(n_1 + 1, n_2) = | x_j^{h}(n_1 + 1, n_2) |, \quad j = 1, \ldots l.
\end{aligned} \tag{13}$$

The same inequality holds for $| x_j^{-v}(n_1, n_2 + 1) |$. For points along the diagonal $n_1 + n_2 = const.$ we now choose:

$$| x_i^{-h,v}(n_1, n_2) | = x_i^{h,v}(n_1, n_2) \tag{14}$$

which is a special case of (10). This automatically provides zero initial conditions for the linear shift-invariant reference system, as zero initial conditions were assumed for the shift-variant system, i.e.

$$| x_i^{-h,v}(0, n_2) | = x_i^{h,v}(0, n_2) = 0$$

$$| x_i^{-h,v}(n_1, 0) | = x_i^{h,v}(n_1, 0) = 0 \tag{15}$$

Choosing the input signal of the linear reference system according to (9) and applying (13) we have for the states of the next diagonal

$$| x_j^{-h}(n_1 + 1, n_2) | \leq x_j^{h}(n_1 + 1, n_2) \tag{16}$$

$$| x_j^{-v}(n_1, n_2 + 1) | \leq x_j^{v}(n_1, n_2 + 1) \tag{17}$$

Hence all the individual components of the states along the diagonal of points (n_1', n_2'), for which

$$n_1' + n_2' = n_1 + n_2 + 1,$$

belonging to the linear shift-invariant model, constitute an upper bound for the corresponding components of the shift-varying system states. If we now choose

$$| u_i^{-}(n_1, n_2) | = u_i(n_1, n_2)$$

for the computation of the states along the next diagonal, equation (10) will still be met as proved by equation (13). Equation (9) and (13) assure that this result holds for all further diagonals.

It remains to prove necessity: Necessity can again be deduced from the linear shift-invariant system in (4-6) belonging to the class of systems described by equations (1,3,7,8) . Hence the linear shift-invariant system to be stable is a necessary condition for stability of the entire set of systems described by (1,3). This concludes the proof.

In the general m-D case the comparison between the states of the m-D shift-invariant system with the shift-invariant system is conducted on the (m-1) -dimensional hyper planes. Except for the states to consist of m differently propagating portions instead of two, the previous proof remains the same. The corresponding results for 1-D systems can be found in [10].

The above proof also shows, that the linear shift-invariant system in (4) constitutes an upper bound system for the shift-varying system in (3) if the initial conditions and inputs of the shift-invariant system are chosen properly and if (7) and (8) are fulfilled. Due to equations (11) and (12) the linear shift-invariant system can be used to estimate the margin of stability of the shift-varying system. In most of the cases, this will again be a conservative estimate.

The stability condition of Theorem 1 can be interpreted as follows:

From the necessary and sufficient condition for stability of the linear shift-invariant system matrix $A = [a_{i,j}]$ with $a_{ij} \geq 0$ it follows, that all system matrices A^- with entries a_{ij}^- varying inside the closed hyper cube

$$\Omega = \{(a_{11}^-, a_{12}^-, \ldots, a_{l+k,l+k}^-) \mid a_{ij}^- \epsilon [-a_{ij}, a_{ij}]\}$$

are also stable.

The robust stability conditions derived for shift-varying state space systems can certainly also be used for shift-invariant system matrices yielding the following Corollary:

Corollary 2:

The shift-invariant interval system matrix $A = [a_{ij}^*]$ with entries inside the closed hypercube:

$$\Omega = \{(a_{11}^*, a_{12}^*, \ldots, a_{l+k,l+k}^*) \mid a_{ij}^* \epsilon [-a_{ij}, a_{ij}]\} \qquad (18)$$

is stable, iff the corner $(a_{11}, a_{12}, \ldots, a_{l+k,l+k})$ consisting of non-negative elements only, is stable.

Corollary 3:

An upper or lower triangular interval system matrix A, (A^-) of a shift-invariant system (shift-variant system) is stable, iff the intervals along the diagonal have boundaries which are strictly smaller than 1 in magnitude.

Proof:

The proof simply follows from the fact, that the positive, shift-invariant reference matrix in (4) is also triangular. A triangular shift-invariant 2-D Roesser matrix is stable, iff all the diagonal elements are strictly smaller than 1 in magnitude. This is due the fact, that a upper or lower triangular Roesser matrix always represents a separable 2-D system, the eigenvalues of which are distinct and are given by the diagonal matrix elements. This concludes the proof.

For system matrices with non-symmetric intervals, Theorem 1 can still be used as a sufficient condition. Assume the non-symmetric matrix entry

$$a_{ij}^{-} \epsilon [c_{ij}, d_{ij}].$$

Theorem 1 can be applied again by observing that

$$\mid a_{ij}^{-} \mid \leq a_{ij}, \quad a_{ij} = max\{\mid c_{ij} \mid, \mid d_{ij} \mid\}.$$

Since the artificially created symmetric intervals are larger than required, the condition is sufficient, but not necessary.

For the 1-D case this condition is equivalent to the robust stability condition stated in [1], which is an implication of the Perron - Frobenius Theorem [2,3]. Hence our general result can be viewed as an extension of this implication to the shift-varying and / or multi-dimensional case.

Theorem 1 also extends the results in [8] and [11], which deal with robust stability of m-D first order difference equations and asymptotic stability of zero-input m-D systems respectively.

III. Examples

Two examples will be presented to ilustrate the results, introduced in Section 2.

Example 1:

Consider the 1h-2v 2-D interval matrix:

$$\left(\begin{array}{c|cc} [-0.2, 0.2] & [-0.2, 0.2] & [-0.5, 0.5] \\ \hline [-0.2, 0.2] & [-0.2, 0.2] & [-0.5, 0.5] \\ [-1, 1] & 0 & 0 \end{array}\right)$$

Since the reference matrix with positive coefficients

$$\left(\begin{array}{c|cc} 0.2 & 0.2 & 0.5 \\ \hline 0.2 & 0.2 & 0.5 \\ 1 & 0 & 0 \end{array}\right)$$

is stable, by Theorem1, the interval matrix is also stable for both cases, the shift-variant and the shift-invariant case.

Example 2:

Consider the upper triangular 2h-2v 2-D interval matrix:

$$\left(\begin{array}{cc:cc} [-0.4,0.9] & [-2,9] & [0,0.8] & [-4,-1] \\ 0 & [-0.5,0.9] & [0,5] & [-3,3] \\ \hdashline 0 & 0 & [-0.8,0.5] & [-2,0] \\ 0 & 0 & 0 & [-0.95,0.2] \end{array}\right)$$

By Corollary 3 this matrix is stable, since all the diagonal intervals are a subset of the interval $(-1,1)$. This is true for the shift-variant and the shift-invariant case.

IV. Conclusion

A method for the analysis of BIBO-stability of 2-D shift-varying digital systems was introduced.

The results can be utilized to analyze BIBO-stability (or global asymptotic stability) of nonlinear discrete systems. Also adaptive and linear robust systems can be treated using this technique. The approach is tailored for systems with many shift-varying thick coefficients, which all have a symmetric range around zero. The presented result constitutes an extension of one implication of the Perron-Frobenius Theorem to the shift-varying and/or multi-dimensional case.

Further research is needed in order to improve stability conditions for coefficient ranges, which are not symmetric. In contrast to the continuous polynomial case, there are only very few results available for discrete shift-invariant interval matrices and polytopes of matrices.

References:

[1] J. Garloff, N. K. Bose, "Boundary implications for stability properties: Present status" in *Reliability in computing*, ed. by R. E. Moore, Academic press 1988.

[2] R. S. Varga, *Matrix iterative analysis*,Prentice Hall, Englewood Cliffs, N.J. 1962.

[3] E. Seneta, *Non-negative matrices*, New York, 1973.

[4] E.I. Jury, "Stability of multi-dimensional systems and related problems" chapter 3 in: *Multi-dimensional systems: Techniques and applications*, ed. by T. Tzafestas, Marcel Dekker Inc, New York, Basel 1986.

[5] E.I. Jury, "Robustness of discrete systems: an overview", 11th World Congress of IFAC, Tallinn, USSR, 1990.

[6] T. Mori and H. Kokame, "A necessary and sufficient condition for stability of linear discrete systems with parameter variation .", *J. of Franklin Institute*, 321, No. 3, pp. 135-138, 1986.

[7] N. K. Bose and E. Zeheb, "Kharitanov's theorem and a stability test of multidimensional digital filters",*IEE Proc.*, 133, pt. 6, No.4, August 1986, pp. 187-190.

[8] P. Bauer, "Robustness and stability properties of first order multi-dimensional

(m-D) discrete processes", *Multi-dimensional Systems and Signal Processing*, Vol.1, pp.75-86, 1990.

[9] R. P. Roesser, " A discrete state space model for linear image processing", *IEEE Transactions on Autom. Control*, vol. AC-20, pp. 1-10, Febr. 1975.

[10] P. Bauer and K. Premaratne, "Robust stability of time-variant interval matrices", to appear at the 1990 CDC, Honolulu, Hawaii.

[11] P. Bauer and E. I. Jury " A stability analysis of two-dimensional (2-D) nonlinear digital state space filters", *IEEE Trans. on Acoust.,Signal and Speech Proc.*, Vol. 38, No. 9, pp. 1578-1587, 1990.

A Space of Non-Bandlimited Signals That Can Be Reconstructed From Non-Uniformly Spaced Samples

Myriam Djebeniani and Douglas Cochran
Department of Electrical Engineering
Arizona State University
Tempe, AZ 85287-5706

Abstract- In 1985, Clark *et al* presented a coordinate transformation method that extends uniform sampling theory to allow the reconstruction of a certain class of non-bandlimited signals from sets of non-uniformly spaced samples. This paper examines the space of signals to which this reconstruction technique applies. It also discusses the construction and properties of the space

Reconstruction Technique

Let B denote the set of bandlimited signals $f(t)$ defined for $-\infty < t < \infty$ and Γ the set of bijective functions $\gamma : R \to R$. Note that each $\gamma \in \Gamma$ is a monotone function and has an inverse γ^{-1} such that $\gamma^{-1}(\gamma(t)) = \gamma(\gamma^{-1}(t)) = t$ for all $t \in (-\infty, \infty)$. If $f(t)$ is a function in B with bandwidth $\omega_0 = \frac{\pi}{T}$ and is sampled at the points $t_n = nT$, the Shannon sampling theorem allows the exact reconstruction of $f(t)$ from its uniformly spaced samples as follows:

$$f(t) = \sum_{n=-\infty}^{\infty} \frac{f(nT)\sin(\omega_0(t-nT))}{\omega_0(t-nT)} \tag{1}$$

This classical result can be extended to allow the reconstruction of certain non-bandlimited signals from non-uniformly spaced samples. For example, let $h(t)$ be the non-uniformly sampled function with the nth sample time t_n not necessarily equal to nT. If one can find a function $f(t) \in B$ with bandwith ω_0 and a function $\gamma(t) \in \Gamma$ with $\gamma(t_n) = nT$ such that $h(t) = f(\gamma(t))$; then by letting $\tau = \gamma(t)$ one obtains a function $f(\tau)$ having uniformly spaced samples. This function can then be exactly reconstructed by using equation (1) and applying the mapping from $f(t)$ back to $h(t)$ to get:

$$h(t) = f(\gamma(t)) = \sum_{n=-\infty}^{\infty} \frac{h(t_n)\sin(\omega_0(\gamma(t)-nT))}{\omega_0(\gamma(t)-nT)} \tag{2}$$

This reconstruction is possible since the samples $h(t_n) = h(\gamma^{-1}(nT)) = f(nT)$ are uniform. Once $f(t)$ is reconstructed from the samples $h(t_n)$, the values of h can be obtained from the expression $h(t) = f(\gamma(t))$.

Space to Which The Reconstruction Technique Applies

It can be seen that the set of functions that can be exactly reconstructed with (2), has elements of the form $f \circ \gamma$ for some $f \in B$ and $\gamma \in \Gamma$. Let us denote this set by H; then, $H = B \circ \Gamma$ where $B \circ \Gamma$ is the set of all signals that can be formed by the composition of a signal in B and a function in Γ.

Using the shifting and scaling properties of the Fourier transform, it is easily seen that the composition of a bandlimited signal and an affine function of the form $\gamma(t) = at + b$ with $a \neq 0$ is bandlimited. Therefore, H is just an extension of the set B of bandlimited signals, and if $\gamma(t) = t$ then $B \circ \gamma \equiv B$.

It is important to note that the functions in H are generally not bandlimited. This is stated in [2] as a conjecture:

Conjecture 1 *If $f \in B$ and $\gamma \in \Gamma$, then the composition $f \circ \gamma$ is in B if and only if γ is a non-constant affine function.*

From the conjecture we can determine that, not only are there many signals in H that are not bandlimited, but that this is the typical situation. There are several approaches to demonstrate this. An example of a non-bandlimited signal produced by the composition of a bandlimited signal and a bijective function is constructed based on the Paley-Wiener Theorem[3].

Example 1 *The function f defined as $f(t) = sinc(t)$ is bandlimited. However, $f(t^n) = sinc(t^n) \notin B$ for any integer $n > 1$ because it is not of exponential type.*

This argument can lead to a more general result stated as a property:

Property 1 *Given $f \in B$, $h(t) = f(t^n) \notin B$ for any integer $n > 1$.*

Although H contains many signals that are not bandlimited, it was shown in [2] that it does not contain every non-bandlimited signal. Therefore, not all non-bandlimited signals can be sampled and reconstructed with the technique described in the previous section. We also show that H does not contain every element of the Hilbert space of finite-energy signals L^2. This can be illustrated by an example based on the property that a bandlimited signal cannot be time limited.

Example 2 *The L^2 signal h_1 with values $h_1(t) = u(-t+\pi/2) - u(-t-\pi/2)$ cannot be expressed in the form $h_1 = f \circ \gamma$ for some $f \in B$ and $\gamma \in \Gamma$. In fact, given any $\gamma \in \Gamma$, the function $h_1 \circ \gamma^{-1}$ is always time limited whether γ^{-1} is increasing or decreasing. Therefore, $h_1 \circ \gamma^{-1} = f \notin B$.*

We extend this result to show that the space H does not contain every continuous L^2 signal . An example of such a signal is given below:

Example 3 *The signal $h_2(t)$ defined as $h_2(t) = \cos(t)h_1(t)$ is a continuous L^2 signal; but $h \notin H$.*

The above results lead to the following property:

Property 2 *If $h \in L^2$ is time limited, then $h \notin H$; i.e., h cannot be expressed as $h = f \circ \gamma$ for some $f \in B$ and $\gamma \in \Gamma$.*

Approximation of L^2 Signals

Although H does not contain every L^2 signal, the Plancherel Theorem [3] implies that the set of signals that are in H is dense in L^2. This means that given $\varphi \in L^2$ and $\epsilon > 0$, there is an $h \in H$ with $||\varphi - h||_2 < \epsilon$. Thus, the set of signals that can be exactly reconstructed from non-uniform samples by this technique is dense in L^2.

For a given L^2 signal, however, several approximations may exist. Therefore, a good representation should be chosen based on some criteria that yield an optimum reconstruction of the signal.

Acknowledgements

This work was partially supported by the ASU Telecommunications Research Center. The authors wish to thank J. Clark for his contribution.

References

[1] J.J. Clark, M.R. Palmer, and P.D. Lawrence, "A Transformation Method for the Reconstruction of Functions from Nonuniformly Spaced Samples," IEEE *Trans.* ASSP, ASSP-33(4), October 1985.

[2] D. Cochran and J.J. Clark, "On the Sampling and Reconstruction of Time Warped Band-Limited Signals," *Proc. ICASSP-90*, Vol.3, pp1539-1541, Albuquerque, April 1990.

[3] W. Rudin, *Real and Complex Analysis*, McGraw-Hill, 1974.

Signal Design-Integral, Differential Equations, Spectral Estimation, Capacity Computation, Binary Random Processes-Zero Crossing Problem

Garimella Rama Murthy*,
Bellcore, NVC, 2X-152,
331 Newman Springs Road,
New Jersey-07701.

Abstract

Continuing on the earlier reported results of the author on *spectraofboundedfunctions*, in this paper, a detailed exposition of the results is made. One of the problems is concerned with the maximization of the output signal energy of a linear filter to an input pulse of bounded support and amplitude. Other problem arises in the maximization of the capacity of a bandlimited Gaussian channel to a peak constrained input and is concerned with the maximization of certain entropic functionals. This investigation naturally provides links between maximum entropy spectral estimation, capacity computation and linear mean square estimation. Furthermore, it also leads to a study of the properties of binary random processes and to specifically, a new characterization of positive definite functions attainable as the autocorrelation function of a binary random process. As a related development an approach to the solution of the important *Zero crossing problem* is provided. An interesting approach to express the Fourier transform of an integrable function as the difference of two infinite series is also presented.

1. Introduction:

Deterministic and stochastic waveforms with bounded amplitude and bounded support arise ubiquitously in various communication systems. In the context of signal design for magnetic and optical recording channels, it is natural to maximize the energy of a filtered version of an amplitude constrained pulse with finite support. A natural analogous question for stochastic waveforms is to determine how close is the power spectral density of a stationary amplitude constrained random process to a flat band limited spectrum. This problem is related to the maximization of an entropic functional associated with the random process and to the problem of meaningful characterization of autocorrelation function of a binary random process. These inquiries naturally lead to links between the concepts of maximum entropy spectral estimation, time series analysis and capacity computation for peak constrained, bandlimited Gaussian channels. This sequence of investigations point towards some basic inquiries related to continuous time Fourier transform. It is specifically shown that the Fourier transform of an L^1 function can be expressed as the difference of two infinite series. The two problems initially mentioned were formulated by Wyner [GoC].

2. Bounded Amplitude, Bounded Support Signal Design: Fixed Points of Differential and Integral Equations: Zero Crossings

In the following, the problem of maximization of the output energy of a linear time invariant filter/channel to an amplitude constrained pulse of finite support is considered. The maximum principle approach to the solution is discussed and the other approaches considered in detail in [Rama] are briefly mentioned.

Maximum Principle Approach:

Utilizing the Parseval's relation, the ouput energy E can be expressed as

$$E = \int_{-\infty}^{\infty} |x(t)*h(t)|^2 dt. = \int_{-\infty}^{\infty} | y(t) |^2 dt. \tag{2.1}$$

Let the class of signals of interest be denoted by X.

Theorem 1: If a signal $x(.) \in X$ is optimal, then

$$x(t) = -\,sign(\mathbf{B}^T \lambda(t)), \tag{2.2}$$

where $sign(.)$ is the signum function and $\lambda(t)$ is a fixed point of the following differential equation.

$$\dot{\lambda}(t) = -\mathbf{A}^T\lambda(t) + \mathbf{C}^T \int_0^t h(t-p)x(p)dp\ , \tag{2.3}$$

where the matrices **A**, **B**, **C** are obtained from the state space description of the linear time invariant filter.

$$\mathbf{Z}^{\cdot}(t) = \mathbf{A}\mathbf{Z}(t) + \mathbf{B}x(t), \tag{2.4}$$

$$y(t) = \mathbf{C}\mathbf{Z}(\mathrm{t}). \tag{2.5}$$

$\lambda(t)$ is also the state vector of the adjoint/conjugate linear system corresponding to the state description given in (2.4) and (2.5). The adjoint system is given by

$$\dot{\lambda}(t) = -\mathbf{A}^T\lambda(t) + \mathbf{C}^T y(t) \tag{2.6}$$

$$x(t) = -\,sign(\mathbf{B}^T\lambda(t)). \tag{2.7}$$

■

Remark 1: The optimal signal is thus necessarily binary in nature. A sufficient condition for a signal that satisfies (2.6) and (2.7) to be optimal is that the second variation of J be negative. This is equivalent to requiring that

We now show that for an $x(.) \in X$ to be optimal, it should be the fixed point of an integral equation.

Theorem 2: If a signal $x(.) \in X$ is optimal, then it is the fixed point of the following integral equation

$$x(t) = sign\left(\int_0^t x(p)\, R(t-p)\, dp \right). \tag{2.8}$$

where

$$R(t) = \int_0^{\infty} h(t+u)\, h(u) du.$$

is the autocorrelation function of a causal linear filter. ■

Brief Numerical Discussion: It is well known that the following is a *Volterra Integral equation* and it is a contraction mapping for every value of λ.

$$f(t) = \lambda\left(\int_0^t f(p)\, K(t,p)\, dp \right).$$

The equation in (2.8) is a version related to the above with the Kernel depending on the difference of the arguments. The solutions of interest to us are binary functions over [0,1]. These numerical issues are discussed in [Rama].

It is shown in [Rama] that the result described in (2.8) is also coherent with the result obtained by upper bounding E_1 through the Cauchy-Schwartz inequality.

Remark: It should be noted that a homogeneous Fredholm integral equation of second kind with a kernal R(.) depending on the difference of the arguments will naturally result when E is upper bounded. It will be a contraction mapping only under very specific conditions on k.

We now state a more general problem than the one considered above and provide a solution to it. The solution subsumes the above as a special case.

Problem : Maximize the output energy of time varying linear filter subject to the condition that the input is bounded in amplitude by 1.

Theorem 3: If a signal $x(.) \in X$ is optimal, then

$$x(t) \;=\; -\,sign(\,\mathbf{B}^T(t)\,\lambda(t)\,), \tag{2.9}$$

where $sign(.)$ is the signum function and $\lambda(t)$ is a fixed point of the following differential equation.

$$\dot{\lambda}(t) = -\mathbf{A}^T(t)\lambda(t) + \mathbf{C}(t)^T \int_0^t h(t-p)x(p)dp \tag{2.10}$$

where the matrices $\mathbf{A}(t)$, $\mathbf{B}(t)$, $\mathbf{C}(t)$ are obtained from the linear time varying system description similar to the one in (2.6) and (2.7).

Furthermore, x(.) is also the fixed point of the following integral equation.

$$x(t) \;=\; sign\,(\int_0^t x(p)\,R\,(t,p)\;dp). \tag{2.11}$$

■

A pleasing physical explanation of the previous results is provided in [Rama]. More so, it is shown that the impulse respone, frequency domain and the function space view points provide new insights and special case results to the signal design problem [Rama].

3. Maximization of Entropic Functionals: Capacity Computation, Spectral Estimation and Least Mean Square Estimation

We now consider a problem which arises in various applications.

Problem: Let x(t) be a real-valued stationary random process with $Ex(t) \equiv 0$ and $|x(t)| \equiv 1$. Maximize

$$Q \triangleq \int_{-1}^{+1} \log(1 + S(f))df, \tag{3.1}$$

where S(f) is the power spectral density of the random process x(t).

■

An upper bound on Q is provided and it is then argued in Theorem 4 that under reasonable practical assumptions on the available data, the problem can be completely resolved. It is then described how the concepts of spectral estimation, capacity computation and the least mean square estimation are interrelated.

$$Q \leq 2 \log (1 + \int_{0}^{1} S(f) df), \tag{3.2}$$

which is the same as

$$Q \leq 2 \log (1 + \frac{1}{2} \int_{-1}^{+1} S(f) df). \tag{3.3}$$

Remark: It should be noted that, since logarithm is a strictly concave, but not a linear function, the above inequality is an equality only when the integrand is a constant on the region of integration. Wyner also provides the above upper bound [CoG].

Direct application of Jensen's inequality on Q also leads to the following bound which is tighter than the above when $\int_{0}^{1} S(f) df \geq 1$.

$$Q = \int_{-1}^{+1} \log (1 + S(f)) df. \tag{3.4}$$

$$Q \leq \log 2(1 + \int_{0}^{1} S(f) df). \tag{3.7}$$

Since $2x \leq x^2$ for $x \geq 2$, it is evident that the bound in (3.7) is smaller than (3.3). It should be noted that the above bound holds for any random process.

From (3.5) and from the condition for equality in Jensen's inequality, we have the following bound on Q.

$$Q \leq 2 \log(1 + \frac{1}{2} \int_{-1}^{1} \frac{A^2}{2} df). \tag{3.8}$$

The upper bound is thus reached only when

$$S(f) = \left\{ \begin{array}{ll} \frac{A^2}{2} & \cdots \ abs(f) < 1 \\ 0 & \cdots \ abs(f) > 1 \end{array} \right\}. \tag{3.9}$$

Since the sync function has the above Fourier transform, the attainability of the upper bound is determined by the proximity of the autocorrelation function of a binary stochastic process to a sync function. The problem of maximization is approached through the calculus of variations in [Rama].

The problem of attainability of upper bound on Q by a binary random process, naturally motivates a discussion of the class of positive definite functions attainable as the autocorrelation function of such processes. This problem is considered in section 4.

The problem of maximum entropy spectral estimation of a second order stationary random process is to maximize

$$\int_{-0.5}^{+0.5} \log(S(f)) df \tag{3.10}$$

subject to the following set of constraints on autocorrelations

$$R(n) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} S(\omega) e^{j\omega n} dw \quad \text{for} \quad |n| \leq p. \tag{3.11}$$

The Maximum entropy approach was widely acclaimed to provide better resolution for spectral estimation than the conventional Fourier based methods. But Van den Bos announced his important discovery that the problem of fitting the celebrated maximum entropy spectrum to the given set of autocorrelations is same as that of fitting an autoregressive model to the data [Van]. In view of the resemblance of the entropic functionals in (3.1) and (3.10), it is important to know if the process which maximizes the functional in (3.1) is Markov. We now maximize the entropic functional in (3.1) subject to a finite set of autocorrelations known from the data.

We clearly remark that we consider the discrete time stochastic process obtained by sampling the original continuous time version. Thus the set of autocorrelation functions of interest to us is a countable set.

Theorem 4: The process maximizing the entropic functional

$$Q \triangleq \frac{1}{2\pi} \int_{-1}^{+1} \log(1 + S(w))dw, \tag{3.12}$$

subject to the following set of constraints on autocorrelations

$$R(n) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} S(\omega) e^{j} \omega n dw \quad \text{for} \quad |n| \leq p. \tag{3.13}$$

is an Autoregressive Moving Average Process.

■

Now, we consider the problems of capacity computation and the linear mean square estimation.

It is well nown that the capacity of a channel with an input having the power spectral density $S_x(f)$ is upper bounded by the capacity, C_G of a channel with Gaussian input possessing the same power spectral density, $S_x(f)$. The capacity, C_G is given by

$$C_G = \frac{1}{2} \int_{-\infty}^{+\infty} \log(1 + \frac{PS_y(f)}{\frac{N_0}{2}} df, \tag{3.14}$$

where $S_y(f)$ is the output power spectral density. Thus the channel capacity expression has the same form as that of Q in (3.1). Consequently, the results on maximization of Q and the previous discussion on maximum entropy spectral estimation can be naturally related to the maximization of the channel capacity. It has been argued by Shitz et al that the upper bound described in (3.8) cannot be achieved and they provide an upper bound. We provide a detailed enquiry on these aspects utilizing the property that the space of autocorrelation functions of binary stochastic processes is convex and compact.

If the autocorrelation function of the input sequence is constrained to assume specific values for finitely many lags, then the process which provides the maximum capacity is an Autoregressive Moving Average Process with a very specific structure for the numerator and denominator parts.

We now relate the maximization of Q to the the linear mean square estimation. Yovits and Jackson [Kai] provide the following closed form expression for the mean square error in estimation of a signal process z(.) in additive white noise ν(.)

$$E[z(t) - \hat{z}(t)]^2 = \int_{-\infty}^{\infty} \log(1 + S_z(iw)dw. \tag{3.15}$$

It is natural to enquire about the structure of the spectral density which will minimize the mean square error when the autocorrelation function is constrained. Theorem 4 provides a clue to the solution.

4. Characterization of Autocorrelation Function of Binary Random Processes:

Now, the problem of characterization of autocorrelation function of a binary random process, called the *unit autocorrelation* is considered.

Previous effort on this related class of important problems is made by Mc Millan [Mcm], Shepp [Shp], Masry [Mas] and Martin De Carvalho et al.

Mc Millan obtained a characterization of the class U of unit autocorrelations as stated in the following Theorem.

Theorem 5: $R \in U$ if and only if $R(0) = 1$ and

$$\sum_{n=1}^{N} \sum_{m=1}^{N} R(t_n - t_m) C_{nm} \geq 0 \tag{4.1}$$

for every integer N, every set of points $\{ t_1, \cdots, t_n \}$, $t_k \in R^1$, and all corner positive matrices $[C_{nm}]$, where an $N \times N$ matrix $[C_{nm}]$ is corner-positive if and only if

$$\sum_{n=1}^{N} \sum_{m=1}^{N} C_{nm} \, e_n \, e_m \geq 0 \tag{4.2}$$

for every sequence $(e_1, \cdots, e_N)$ with $e_i = \pm 1$, i=1,2,...,N.

■

This Theorem was proved by Shepp who also provided various important properties of the unit autocorrelations. Functions satisfying the properties provided in the above definition are coined *corner positive definite* functions.

To aid a new approach to the problem, insights into the characterization of corner positive matrices and functions are provided in [Rama].

We now propose an approach which unifies all the results and also provides some new results. The following Theorem relates the autocorrelation functions to the characteristic functions.

Theorem 6: A positive definite function is an autocorrelation function of a stationary binary stochastic process if and only if it is the characteristic function of a counting process evaluated at π. Alternatively a positive definite function is the autocorrelation of a binary stochastic process whose underlying point process has stationary independent increments, if and only if it is the two dimensional characteristic function of a binary state space random process evaluated at $(\frac{\pi}{2}, \frac{3\pi}{2})$.

Proof:

■

The following proposition enables us to derive the properties of unit autocorrelations from those of the chracterisitic functions.

Proposition: If R(.) is the autocorrelation function of a binary random process, then it is the characteristic function of a symmetric random variable.

Proof:

■

The autocorrelation function, by virtue of being the characteristic function of a random variable satisfies the properties of symmetry, uniform continuity, convexity, closure under finite products. Proofs can be found in [Chung, pp.143].

Now it is described how some of these properties have natural interpretations in terms of the underlying point process.

By the Theorem 3.1, $R(t) = E[\, e^{j\pi N(t)} \,]$. But if a binary random process is generated by the superposition of finitely many independent random processes, then the underlying point process can be considered to be the superposition of finitely many random processes. Hence on utilizing $N(t) = \sum_{n=1}^{N} N_i(t)$, the property (iv) follows immediately. Now an infinite version of this property roughly shows that the superposition of large number of low bit rate binary random processes behaves as a binary random process whose underlying point process is Poisson. Formally it is proven in the following limit Theorem and its implications to digital communications are discussed in the sequel.

Theorem 7:: The superposition of a large number of uniformly sparse, mutually independent binary random processes converges to a binary stochastic process whose underlying point process is Poisson. The binary random processes $\{ X_{iK},\ 1 \le i \le K \}$ are said to be uniformly sparse if the underlying point processes are such that

$$\lim_{k \ge \infty} \operatorname*{supremum}_{1 \le i \le k} P\,\{ N_{ik}(t) \ge 1 \} = 0$$

■

Thus, it can be noted that the properties derived in [Shepp] follow directly from the equivalence in Proposition.

Convergence Properties:

The proposition enables us to invoke the convergence Theorems of Levy and Cramer on characteristic functions to deduce the convergence properties of unit autocorrelations.

This approach thus effectively provides a way to determine the class of characteristic functions which can be the autocorrelation of a binary stochastic process. Specifically the problem reduces to determining the *subclass* of characteristic functions which are *corner positive definite* in the spirit of Theorem 5. In this direction, various properties of this class of characteristic functions are now described.

Theorem 9: If f is a unit covariance, then so is $e^{\lambda (f-1)}$ for each $\lambda \ge 0$.

Proof:

■

Theorem 10: A characteristic function corresponding to the unit covariance is that of a lattice distribution if and only if there exists a $t_0 \ne 0$ such that $|\, R(t_0) \,| = 1$. Furthermore if $R(t_0) = R(0)$ for some $t = t_0 \ne 0$, then R(.) is periodic with period t_0.

We now consider the binary stochastic processes in which the underlying point process has a specific structure such as independent increments or I.I.D. interarrival times.

Proposition: If N(.) has independent incrementnts, then the autocorrelation function corresponds to the characteristic function of Cauchy random variable.

5. Zero Crossing Problem:

This problem is concerned with the determination of *the distribution of number of zero crossings of a stochastic process in an interval.* This problem arises naturally in various applications in physical science including hydrology, seismology, meteorology, reliability theory, storage theory, medicine, aerodynamics and structural engineering [Abr]. WE now provide an approach to its solution.

Zero Crossing Problem-Unit Autocorrelation Characeterization:

The relationship between the characterization problem and the Z.C.P is now briefly described.

The autocorrelation function of a stationary binary random process is given by

$$R(\gamma) = P\{X(t) = X(t+\gamma)\} - P\{X(t) \neq X(t+\gamma)\}. \qquad (5.1)$$

where $X(t) = X(0)(-1)^{N(t)}$.

$$= P\{N(\gamma)\ is\ even\} - P\{N(\gamma)\ is\ odd\}. \qquad (5.2)$$

An expression for the utocorrelation function is given by

$$= \sum_{n=0}^{\infty} (-1)^n P(N(\gamma) = n) \qquad (5.3)$$

Since the zero crossing problem essentially requires the determination of $P\{N(\gamma) = n\}$, for *all* $n \geq 0$, it is evident that the problem of characterization of the autocorrelation function of a stationary binary stochastic process and the Zero Crossing problem, essentially require some characterization of the associated characteristic function.

Expanding R(t), a characeteristic function, using the moments,

$$R(t) = \sum_{n=0}^{\infty} \frac{m_n}{n!} (jt)^n. \qquad (5.4)$$

An expression for R(t) is also given by,

$$= \sum_{n=0}^{\infty} (-1)^n P(N(t) = n). \qquad (5.5)$$

Utilizing the fact that the odd moments of the underlying random variable/probability measure should all be zero for a real characteristic function (corresponding to a symmetric random variable),

$$= \sum_{n=1}^{\infty} (-1)^n m^{(2n)} \frac{t^{2n}}{2n!}. \qquad (5.6)$$

This reduction, on equating the coefficients, effectively provides an answer to the *ZeroCrossingProblem*. THUS

$$P\{N(t) = n\} = \frac{m^{(2n)}\, t^{2n}}{2n!}$$

From the discussion preceeding the solution for Markov processes, it is evident that the *zero crossing problem* of interest in various applications, can be resolved once the even moments of the characteristic function associated with a binary random process, obtained by thresholding the original process, can be computed. Since the even moments are basically the even order derivatives of the characteristic function or equivalently the autocorrelation function, it is now shown how the zero crossing problem can be resolved when the input process to the hard limiter is a normal random process.

Normal Random Processes:

It is well known that the autocorrelation function of the output of a hard limiter driven by a normal process is given by

$$R_Y(t) = \frac{2}{\pi} Sin^{-1} \frac{R_x(t)}{R_x(0)}. \tag{5.8}$$

Now the even order derivatives of this can be computed and the zero crossing problem can in principle be resolved.

Arbitrary Stationary Stochastic Processes:

Now we come to the last case where the input process to the hard limiter is neither *MarkovnorGaussian*. We have two cases.

Case A: The time intervals between successive zero crossings forms a renewal process.

Utilizing the result in [Mas], the general expression for the Laplace transform of the autocorrelation function is given by

$$f(s) = \frac{1}{s} - \frac{2\beta}{s^2} + \frac{4\beta}{s} \frac{\phi(s)}{1+\phi(s)}, \tag{5.9}$$

where *phi* (s) is the characteristic function of a distribution F(x) on $[0, \infty)$ with a finite mean $\frac{1}{\beta}$.

Now using the general expression for the n th derivative and using the initial value Theorem, the even order derivatives at the origin of the autocorrelation function can be computed. Thus the zero crossing problem in this case can be resolved.

Consider a renewal process in which the interarrival distribution is of phase type. Then using the results in [Mas], an explicit expression for the unit autocorrelation function is computed in [Rama2]. Since any distribution can be arbitrarily closely approximated by a phase type distribution, (in the sense of denseness) this approach is useful.

Case B: Inter zerocrossing times of the stochastic process under consideration forms an arbitrary stationary point process.

From [Mas], expression for the autocorrelation function is given by

$$R(t) = 1 - 2\beta |t| + 4\beta \int_0^{|t|} K(x)\, dx, \tag{5.10}$$

where

$$K(x) = \sum_{n=1}^{\infty} (-1)^n F_n(x). \tag{5.11}$$

From the above, the expression for successive even order derivatives is given by

$$R^{(2n)}(0) = 4\,\beta\,K^{(2n-1)}(0), \tag{5.12}$$

where

$$K^{(2n-1)}(0) = \sum_{i=1}^{\infty} (-1)^i\, F_i^{(2n-1)}(x).$$

Thus the celebrated *Zero Crossing Problem* can be completely resolved under some conditions on existence of moments/derivatives of arbitrary order.

In the case of finite state space Markov processes, the autocorrelation function can be explicitly computed [Rama2]. An expression, in the case of binary Markov processes is provided in [Rama1]. The following proposition is an extremely important one, which has some very fundamental repercussions.

Proposition: No finite state space Markov process is bandlimited.

■

I conjecture that no finite state space stochastic process is bandlimited. Details of this *uncertainity principle* (a process cannot have a bounded spectrum and a countable state space) are provided in [Rama2] where a detailed investigation of finite state space stochastic processes is made.

7. Continuous Fourier Transform: Infinite Series Expansion:

Now we focus our attention on an interesting idea for expressing the continuous Fourier transform of function as a difference of two infinite series. This representation might prove to be very useful from a theoritical and applied stand point.

Theorem 4: The Fourier transform of an integrable function f can be represented as the difference of two infinite series.

References:

[Abr] J. Abrahams, "A Survey of Recent Progress on Level Crossing Problems for Random Processes," in Recent Advances in Communications and Networks, edited by V. Poor et al.

[CoG] T. M. Cover and B. Gopinath, *Open Problems in Communication and Computation,* Springer-Verlag, 1987.

[Mas] E. Masry, "On Covariance Functions of Unit Processes," *SIAM Journal of Applied Mathematics,* vol. 23, no. 1, July 1972.

[Kai] T. Kailath, "A view of three decades of linear filtering theory," *IEEE Transactions on Information Theory,"* pp. 146-181, March 1974.

[KoF] A.N.Kolmogorov and S.V.Fomin, "Introductory Real Analysis," Translation, Dover Publications, Inc.

[MaC] J. L. Martins de Carvalho and J. M. C. Clark, "Characterizing the autocorrelations of binary sequences," *IEEE Transactions on Information Theory,* Vol. IT-29, pp. 502-508, 1983.

[Rama1] G. Rama Murthy, "Spectra of bounded functions," Presented at the Princeton Conference on Information Sciences and Systems, March 1990.

[Rama2] G. Rama Murthy, *"Spectra of bounded functions and applications in communication,"* Manuscript in Preparation.

[Van] A. Van den Bos, "Alternative Interpretation of Maximum Entropy Spectral Analysis," *IEEE Transactions on Information Theory,* vol. IT-17, pp. 493-494, July, 1971.

THE APPLICATION OF PROJECTION ONTO CONVEX SETS (POCS) TO MULTIPATH CHANNEL ESTIMATION[*]

ZORAN KOSTIC[†], EDWARD L. TITLEBAUM[†] and M. IBRAHIM SEZAN[‡]
[†] University of Rochester, Electrical Engineering Dept., Rochester, New York 14627.
[‡] Imaging Information Systems, Eastman Kodak Company, Rochester, NY 14650–1816

THEORY

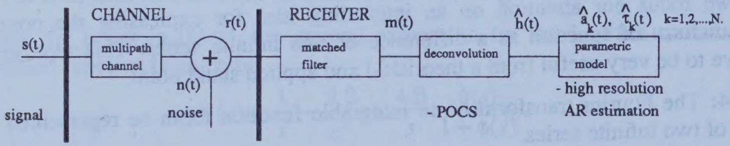
We address the problem of estimating the parameters of specular multipath communication channels. We develop a new method that makes use of matched filtering, POCS deconvolution and autoregressive (AR) modeling. The proposed method is capable of providing good estimates at low signal to noise ratios with modest computational cost. As probing signals, we use pulse compression waveforms such as the chirp, and frequency-hopping waveforms [1].

If we denote the probing signal waveform in a specular multipath communication channel as *s(t)* then the received signal is

$$r(t) = s(t) \otimes h(t) + n(t) = \sum_{k=1}^{N} a_k s(t - \tau_k) + n(t)$$

where *h(t)* is the channel impulse response and *n(t)* denotes the channel noise. The multipath estimation problem refers to estimating the parameters a_k (path attenuations), τ_k (arrival times) and N (number of signal paths).

A block diagram of the channel and the proposed receiver is shown in the following figure. The output of the matched filter, *m(t)*, is the convolution of the known autocorrelation function *a(t)* of the probing signal *s(t)* and the channel impulse response *h(t)*, plus the convolution of the matched filter impulse response with the channel noise. The effective SNR is increased at the output of the matched filter. In the next stage, the POCS deconvolution of the signal *m(t)* is performed to obtain a high resolution estimate of the channel response *h(t)*. The final stage uses Prony type AR modelling of the frequency domain representation of the channel impulse response to possibly improve the resolution further, and also to obtain the numerical values of the multipath parameter estimates. A similar structure for the estimation of the multipath parameters was proposed by Palas et al. [2], where the deconvolution was performed by inverse filtering. Note that the use of pulse compression waveforms as probing signals in the proposed method results in an autocorrelation function with a narrow main peak and low side lobes, thus facilitating effective deconvolution.



The method of POCS is an iterative technique that finds a feasible solution consistent with a number of *a priori* constraints [3]. It has been successfully applied to various deconvolution problems [4, 5]. In the context of deconvolution using POCS, we have used the bounded residual variance, bounded amplitude, predetermined support and real-valuedness constraint. The bound on the variance of the residual is obtained from the variance of the channel noise which can be measured *a priori*. The amplitude bounds on the channel impulse response are determined from the fact that $0 \leq a_k \leq 1$. The support is estimated from the output of the matched filter. The real-valuedness constraint forces the channel response to be real-valued. Denoting the projection operators projecting onto the convex sets associated with these constraints as P_v, P_b, P_s and P_{re}, respectively, the POCS algorithm is expressed as $\widehat{\mathbf{h}}_{k+1} = P_v P_s P_{re} P_b \widehat{\mathbf{h}}_k$, $k = 0, 1, \ldots$ where $\widehat{\mathbf{h}}_k$ denotes the estimate obtained at the k-th iteration of the algorithm. The signal obtained at the output of the matched filter is used to initialize the POCS algorithm. The details about the procedure and the results obtained can be found in [6].

---

[*] This work was supported in part by the grant from SDIO/IST and managed by the Office of Naval Research under contract N0014–86–K–0511

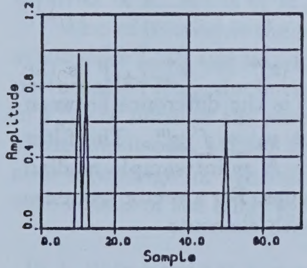
SIMULATION RESULTS

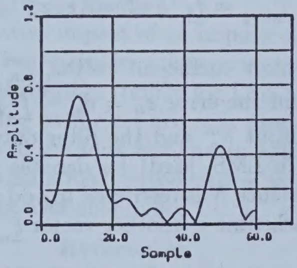
We performed simulations using probing signals with time-radian bandwidth (TB) products ranging from 200 to 500, with the fixed discrete bandwidth equal to 1. We used complex pulse-compression waveforms: chirp, QC coded and Costas coded frequency hopping signals [1]. We simulated the channel noise with zero-mean white Gaussian noise, at different input SNR levels ranging from 0 dB to 40 dB. We expressed the quality of results in terms of the resolution of closely spaced multipath components, and the mean square error in the estimates of the path delays and amplitudes.

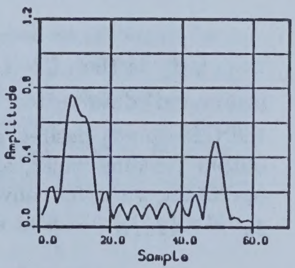
To further evaluate our method, we investigated the issues of the dependence of the estimates on the choice of constraint sets and the noise influence on the estimation. The most effective constraint sets in our simulation turned out to be the bounded residual variance and amplitude constraint sets. The iterations converged rapidly within 10 iterations. Very good parameter estimates were obtained at the end of POCS deconvolution at SNR levels of 5dB and higher. At lower SNR's, spurious low-amplitude path occurred.

We compared our results with those obtained using matched filtering only, matched filtering followed by inverse filtering and matched filtering followed by truncated inverse filtering. We further modeled the deconvolution result of a POCS deconvolution by an AR model, and compared the results to those obtained by previously proposed methods that utilize AR modeling. The results show that the proposed method based on POCS consistently achieves higher multipath component resolution at SNR's where other methods completely fail to work. The proposed method is computationally inexpensive and is easy to implement.

Below we compare the result obtained by POCS method to matched filtering and truncated inverse filtering after matched filtering, for a single three-component multipath model. The SNR is 15 dB, and 10 POCS iterations are performed. The TB product equals 450, and the chirp probing signal is used. The channel model is represented by three ordered pairs in which the first number represents the path delay and the second represents the attenuation of each component: (10,1.0) (12,0.9) (50,0.5). Notice the first two closely spaced paths, which are resolved only with the proposed method.







[1] E. Titlebaum, J. Bellegarda, and S. Maric, "Ambiguity properties of quadratic congruential coding," IEEE Trans. Aerospace and Electronic Systems, to be published 1990.

[2] M. Palas, N. Martin, and J. Martin, "Time delay estimation by autoregressive modelization," in Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, (Dallas), pp. 455–459, 1987.

[3] D. Youla and H. Webb, "Image restoration by the method of convex projections part 1 - theory," IEEE Trans. Medical Imaging, vol. MI, pp. 81–94, October 1982.

[4] H. Trussel and M. Civanlar, "The feasible solution in signal restoration," IEEE Trans. Acoustic, Speech and Signal Processing, vol. ASSP 32, pp. 201–212, April 1984.

[5] M. Sezan and A. Tekalp, "Adaptive image restoration with artifact suppression using the theory of convex projections," IEEE Trans. Acoustic, Speech and Signal Processing, vol. ASSP-38, pp. 181–185, Jan. 1990.

[6] Z. Kostic, "The estimation of multipath parameters using the method of projection onto convex sets," Technical Report EE-90-04, University of Rochester Electrical Engineering Department, June 1990.

Analysis of the Median LMS Adaptive Filter

Peter M. Clarkson * Geoffrey A. Williamson

Dept. of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago IL 60616

Adaptive filtering provides a powerful approach to many signal processing problems. The most widely used adaptive filter is the Least Mean Squares (LMS) algorithm [1]. Although this filter suffers from slow and generally non-uniform convergence, it is famous for its simplicity both in concept and implementation. The performance of gradient search algorithms such as LMS may degrade badly, however, when the filter is subjected to input signals which are corrupted by impulsive interference. Sparse impulses arise frequently in a variety of practical situations, including speech and image processing, as well as biomedical and communications applications. One possible treatment for impulsive corruption is processing the data using nonlinear transformations such as median or other Order Statistic (OS) filters [2]-[3].

An approach which incorporates the attractive robustness of the median operation into an adaptive framework is provided by the median LMS (MLMS) algorithm [4]. This is an adaptive filter similar in structure to the LMS, but which is designed to protect the filter coefficients from the impact of impulsive interferences by applying a median filtering operation to the raw gradient estimates. This is achieved using a modified update equation

$$\underline{f}_{n+1} = \underline{f}_n + \mu \mathrm{Med}\{e_n \underline{x}_n^m\}_N . \tag{1}$$

In (1), $\underline{f}_n$ is the $(L \times 1)$ filter coefficient vector, $\underline{x}_n^m = [x_n^m \ \cdots \ x_{n-L+1}^m]^t$ is the (measured) data vector, and the error $e_n = d_n^m - \underline{f}_n^t \underline{x}_n^m$ is the difference between the (measured) desired output d_n^m and the filter output $y_n = \underline{f}_n^t \underline{x}_n^m$. The filter output remains linear, as in LMS. $\mathrm{Med}\{\cdot\}_N$ denotes the N point sample median operation, where for convenience N is restricted to odd values. For a vector sequence $\underline{s}_n$, $\mathrm{Med}\{\underline{s}_n\}_N$ yields at each time n another vector $\underline{\xi}_n$ with elements

$$\xi_n(i) = \mathrm{median}\{s_n(i), s_{n-1}(i), \ldots, s_{n-N+1}(i)\}. \tag{2}$$

Note that as the linear structure of the filtering operation is preserved, so also is the choice of optimal filter coefficients $\underline{f}^*$ minimizing the mean squared error [1]. Simulations conducted using various forms of impulsive noise have demonstrated the clear, and in many cases decisive, advantages of MLMS compared to LMS or other linear gradient estimators [4]-[5].

---

* Supported in part by National Science Foundation grant no. MIP-8911676.

We have analyzed the mean convergence of MLMS, and compared the results to LMS, subject to both of the following assumptions:

(a.) The base signals d_n and x_n are zero mean iid, are drawn from symmetric densities, and are related by $d_n = \underline{f}^{*t}\underline{x}_n$. The variance of x_n is denoted σ_x^2.

(b.) Impulsive noise enters as $d_n^m = d_n + \eta_n$ and $x_n^m = x_n + \zeta_n$. With fixed, small probability, η_n and ζ_n take nonzero values, and are otherwise zero. The nonzero values are distributed symmetrically with variance large compared to σ_x^2.

These assumptions render tractable an analysis which is in general difficult, due to the incorporation of the nonlinear median operation into the LMS update, itself a nonlinear recursive combination of data vectors. Three cases have been considered in the analysis: (I) $\eta_n \equiv \zeta_n \equiv 0$ (non-impulsive), (II) $\zeta_n \equiv 0$ (impulsive corruption only of d_n), and (III) $\eta_n \equiv 0$ (impulsive corruption only of x_n).

For LMS it is well-known that for case (I) we have exponential convergence of $E\{\underline{f}_n\}$ to $\underline{f}^*$. In case (II), standard techniques [1] show that impulses in the desired signal do not influence the mean convergence to $\underline{f}^*$, though the mean square error at convergence is increased. However, impulses in the input signal (case (III)) *bias* the convergence point away from $\underline{f}^*$, and they also affect the convergence rate.

In contrast, MLMS converges exponentially to $\underline{f}^*$ in all three cases [6]. Moreover, the convergence rate depends on the statistics of $\underline{x}_n$ but unlike LMS shows little dependence on the impulse statistics. Specifically, we have

$$(E\{\underline{f}_{n+1}\} - \underline{f}^*) = (1 - \mu\gamma_n^2)(E\{\underline{f}_n\} - \underline{f}^*), \tag{3}$$

indicating component-wise decoupling (see [6] for details). The quantity γ_n^2 in (3) is bounded by $0 < \beta_1^2 \leq \gamma_n^2 \leq \beta_N^2 < \infty$, where β_i^2 denotes the expected value of the ith order statistic of x_n^2. A good estimate of γ_n^2 which predicts well the simulated behavior of MLMS is $\gamma_n^2 \approx \beta_M^2 = E\{Med\{x_n^2\}_N\}$.

Also of interest is the relative impact of an impulse arrival on the steady-state solution for LMS and MLMS. For sparse impulses restricted to d_n^m, MLMS suffers no perturbation from the optimal solution (unlike for LMS). For an impulse occuring in x_n^m, the norm of the resulting coefficient errors is proportional to the amplitude of the impulse for MLMS and to the square of the amplitude for LMS, indicating a clear advantage for MLMS. Simulations corroborating these observations, together with details of the analysis, are available in [6].

References

[1] B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, 1985.

[2] J.W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, 1977.

[3] A.C. Bovik, T.S. Huang, and D.C. Munson, "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-31, pp. 1342-1349, 1983.

[4] P.M. Clarkson and T.I. Haweel, "A median LMS algorithm," *Electr. Lett.*, vol. 25, pp. 520-522, 1989.

[5] T.I. Haweel and P.M. Clarkson, "Analysis and generalization of a median LMS algorithm," *Proc. ICASSP*, Albuquerque NM, April 1990, pp. 1269-1272.

[6] G.A. Williamson, P.M. Clarkson, and W.A. Sethares, "On the performance of the median LMS adaptive filter," submitted to *IEEE Trans. Acoust., Speech and Signal Processing*, September 1990.

IDENTIFICATION AND ESTIMATION OF MULTICHANNEL NON-GAUSSIAN MOVING-AVERAGE PROCESSES USING FOURTH-ORDER CUMULANTS

YUJIRO INOUYE†
Dept. of Control Engineering
Osaka University
Toyonaka, Osaka 560
JAPAN

LANG TONG and RUEYWEN LIU
Dept. of Electrical Engineering
University of Notre Dame
Notre Dame, IN46556
U.S.A.

Abstract - In this paper, a complete solution for the identification of multichannel moving-average processes using the fourth-order cumulants is presented for the first time. This includes the case of non-minimum phase. A computationally efficient and reliable algorithm is presented to obtain complete estimates of MA coefficient matrices.

I. INTRODUCTION

Parametric modeling of multichannel time series is important in applications such as multivariate linear prediction, spectral estimation ([MA83], [ST77], [IN84], [IN85]), econometrics [HA76], array and seismic data processing [RO67].

Conventional methods for the parameter estimation of multichannel autoregressive and moving-average (ARMA) models based on second-order statistics invoke the assumption that they are of minimum phase ([HA70], [HA76], [PH78]). In order to obviate the minimum phase assumption, we use *cumulants*, which are phase-sensitive higher-order statistics ([KE77], [BR75], [ROs85]). Recently, cumulants have received a revived interest in nonlinear system identification [BR90], phase coupling detection [RA85], and parameter estimation of non-Gaussian univariate ARMA processes ([GI89a], [GI90], [LI90]). In the case of multivariate non-Gaussian processes, third-order cumulant-based methods have been reported for estimating the parameters of AR models by Raguhveer [RA86], and MA models by Inouye *et al.* ([IN86], [GI89], [TO90b]). Third-order cumulants, however, vanish for random variables having symmetric distributions. In many applications, the underlying random processes have symmetric or nearly symmetric distributions. In such a case, third-order cumulants do not convey sufficient information on underlying processes. The previous researches ([IN88], [GI89], [TO90b]) are confined

---

[†] Dr. Inouye was on leave at the University of Notre Dame.

to the third-order cumulant case for brevity of discussions, but present a possibility of extension to the higher-order cumulant case. Recently, fourth-order cumulants have been received a great interest in the source separation problem referred to as the *Blind Identification* of independent components ([CA89a], [CA89b], [TO89], [CA90], [RU89] and [CO89]).

In this paper, we first extend the previous main results ([IN88],[GI89]) to the fourth-order cumulant case. We shall show that an equivalent class of multichannel MA models (driven by mutually independent input sequences whose components are independent, and identically distributed (i.i.d)) that can be identified using the fourth order cumulants is different from that which can be identified using the third-order cumulants. With the fourth-order cumulant-based identifications, we are able to simplify, for many practical cases, the *FInite-Step Global Convergence* (FSGC) algorithm proposed in [TO90b] for the third-order cumulant-based identification. The algorithm obtained here avoids the iterative steps necessary in the FSGC. Hence, the identification of MA model using fourth-order cumulants is computationally simpler than that using third-order cumulants. Finally, simulation results are included to examine the effectiveness of our proposed method.

II. IDENTIFICATION OF MA PARAMETERS

Let $\{\mathbf{y}(n)\}$ be a stationary r-variate random process. We used the third-order cumulants of $\{\mathbf{y}(n)\}$ in the previous researches ([IN88], [GI89], [TO90b]), which are defined as the r×r matrices

$$\mathbf{C}_{yj}(m_1,m_2) = E\{\mathbf{y}(i+m_1)\mathbf{y}^t(i)y_j(i+m_2)\},\ j = 1, 2, \ldots, r,$$

where $y_j(i)$ denotes the *j*th component of the vector $\mathbf{y}(i)$. In this paper, we use the fourth-order cumulants of $\{\mathbf{y}(n)\}$, which are defined as

$$\begin{aligned}\mathbf{C}_{yjk}(m_1,m_2,m_3) = {} & E\{\mathbf{y}(i+m_1)\mathbf{y}^t(i)y_j(i+m_2)y_k(i+m_3)\} \\ & - E\{\mathbf{y}(i+m_1)\mathbf{y}^t(i)\}E\{y_j(i+m_2)y_k(i+m_3)\} \\ & - E\{\mathbf{y}(i+m_1)y_j(i+m_2)\}E\{\mathbf{y}^t(i)y_k(i+m_3)\} \\ & - E\{\mathbf{y}(i+m_1)y_k(i+m_3)\}E\{\mathbf{y}^t(i)y_j(i+m_2)\}, \qquad j,k = 1, \ldots, r. \quad (1)\end{aligned}$$

Assume $\{\mathbf{y}(n)\}$ is a non-Gaussian MA(q) vector process given by

$$\mathbf{y}(n) = \sum_{k=0}^{q}\mathbf{H}(k)\mathbf{w}(n-k) + \mathbf{v}(n), \tag{2}$$

where $\{\mathbf{w}(n)\}$, $\{\mathbf{v}(n)\}$, $\mathbf{H}(0)$ and $\mathbf{H}(q)$ satisfy the following assumption:

A1: $\{\mathbf{w}(n)\}$ is an s-variate, zero-mean, (fourth-order) stationary, and non-Gaussian process with the fourth-order cumulants satisfying

$$\mathbf{C}_{wjk}(m_1,m_2,m_3) = \mathbf{E}_j,\ \text{if } m_1=m_2=m_3,\ j = k,\ j \le \tau,$$

$$= -\mathbf{E}_j, \text{ if } m_1=m_2=m_3,\ j = k,\ j > \tau,$$

$$= \mathbf{0}, \text{ otherwise,}$$

where $\mathbf{C}_{wjk}(m_1,m_2,m_3)$'s are the fourth-order cumulants of $\{\mathbf{w}(n)\}$ defined by (1), and $\mathbf{E}_j = \text{diag}[e_{11}, ..., e_{ss}]$ with $e_{jj} = 1$ and $e_{ii} = 0$ for $i \neq j$. Here $\text{diag}[a_1, ..., a_s]$ stands for an s×s diagonal matrix with diagonal elements $\{a_i\}_{i=1}^{s}$.

A2: $\{\mathbf{v}(n)\}$ is an r-variate, Gaussian, perhaps colored, zero-mean process independent of $\{\mathbf{w}(n)\}$.

A3: $\mathbf{H}(0)$ and $\mathbf{H}(q)$ are of full rank s ($s \leq r$).

Remark. A1 can be understood as $\{\mathbf{w}(n)\}$ being an i.i.d vector process having the components $\{w_i(n)\}$, whose fourth-order cumulants $\kappa_i^{(4)}$ are either positive or negative. Without loss of generality, it is assumed here that

$$\kappa_i^{(4)} = 1 \quad \text{if } i = 1, ..., \tau,$$

$$\kappa_i^{(4)} = -1 \text{ if } i = \tau+1, ..., s.$$

Substituting (2) into (1) and using *A1-A2*, we obtain

$$\mathbf{C}_{yjk}(m_1,m_2,m_3) = \sum_{i=0}^{q} \mathbf{H}(i+m_1)\text{diag}[h_{j1}(i+m_2)h_{k1}(i+m_3),...,h_{js}(i+m_2)h_{ks}(i+m_3)]\mathbf{J}\mathbf{H}^t(i),$$

$$\text{if } 0 \leq m_1,m_2,m_3 \leq q$$

$$= 0, \qquad \text{otherwise,} \tag{3}$$

where $h_{jk}(i)$ denotes the (j,k)th element of $\mathbf{H}(i)$, $\mathbf{J}$ is a diagonal matrix whose first τ diagonal elements are 1 while the last $(s-\tau)$ diagonal elements are -1. Later we shall find that, in order to determine the MA parameters, it is sufficient to consider (3) in case $m_2 = m_3$. Define $\mathbf{C}_{yij}(m_1,m_2)$ as

$$\mathbf{C}_{yij}(m_1,m_2) = \mathbf{C}_{yij}(m_1,m_2,m_2). \tag{4}$$

Then (3) becomes

$$\mathbf{C}_{yij}(m_1,m_2) = \sum_{i=0}^{q} \mathbf{H}(i+m_1)\text{diag}[h_{j1}(i+m_2)h_{i1}(i+m_2),...,h_{js}(i+m_2)h_{is}(i+m_2)]\mathbf{J}\mathbf{H}^t(i),$$

$$\text{if } 0 \leq m_1,m_2 \leq q$$

$$= 0, \qquad \text{otherwise.} \tag{5}$$

Consider (5) with $m_1 = k$, $m_2 = q$, $m_1 = 0$, $m_2 = q$, and $m_1 = q$, $m_2 = 0$, we obtain

$$\mathbf{C}_{yij}(k,q) = \mathbf{H}(k)\text{diag}[h_{i1}(q)h_{j1}(q)...h_{is}(q)h_{js}(q)]\mathbf{J}\mathbf{H}^t(0), \tag{6a}$$

$$\mathbf{C}_{yij}(0,q) = \mathbf{H}(0)\text{diag}[h_{i1}(q)h_{j1}(q)...h_{is}(q)h_{js}(q)]\mathbf{J}\mathbf{H}^t(0), \tag{6b}$$

$$\mathbf{C}_{yij}(q,0) = \mathbf{H}(q)\text{diag}[h_{i1}(0)h_{j1}(0)...h_{is}(0)h_{js}(0)]\mathbf{J}\mathbf{H}^t(0). \tag{6c}$$

Premultiplying (6b) by $\mathbf{H}(k)[\mathbf{H}^t(0)\mathbf{H}(0)]^{-1}\mathbf{H}^t(0)$, and substituting the result into (6a) with $i = j$, we find

$$\mathbf{C}_{yjj}(k,q) = \mathbf{H}(k)[\mathbf{H}^t(0)\mathbf{H}(0)]^{-1}\mathbf{H}^t(0)\mathbf{C}_{yjj}(0,q),\ j = 0, ..., r. \tag{7}$$

Then we define the $r \times r^2$ matrix $\mathbf{L}(k,q)$ as

$$\mathbf{L}(k,q) \equiv [\mathbf{C}_{y11}(k,q), \ldots, \mathbf{C}_{yrr}(k,q)] \tag{8}$$

and use (7) to obtain

$$\mathbf{L}(k,q) = \mathbf{H}(k)[\mathbf{H}^t(0)\mathbf{H}(0)]^{-1}\mathbf{H}^t(0)\mathbf{L}(0,q). \tag{9}$$

Substituting (6b) into (8), we find that $\mathbf{L}(0,q)$ can be rewritten as

$$\mathbf{L}(0,q) = \mathbf{H}(0)\mathbf{M}(q)\mathrm{diag}[\mathbf{J}\mathbf{H}^t(0)], \tag{10}$$

where $\mathbf{M}(q) = [\mathrm{diag}[h_{11}^2(q) \ldots h_{1s}^2(q)], \ldots, \mathrm{diag}[h_{r1}^2(q) \ldots h_{rs}^2(q)]]$ and $\mathrm{diag}[\mathbf{J}\mathbf{H}^t(0)]$ is a block diagonal $rs \times s^2$ matrix with the same diagonal block elements $\mathbf{J}\mathbf{H}^t(0)$. It is then easily shown from *A3* that $\mathbf{M}(q)$ has full row rank s. This means that $\mathbf{M}(q)\mathrm{diag}[\mathbf{J}\mathbf{H}^t(0)]$ is of full row rank, because $\mathrm{diag}[\mathbf{J}\mathbf{H}^t(0)]$ is of full row rank. Therefore, (10) implies from *A3* that $\mathbf{L}(0,q)$ is of rank s. Thus $\mathbf{L}(0,q)\mathbf{L}^t(0,q)$ becomes nonsingular in case of $s=r$. In this case, solving (9) for $\mathbf{H}(k)$, we obtain

$$\mathbf{H}(k) = \mathbf{L}(k,q)\mathbf{L}^t(0,q)[\mathbf{L}(0,q)\mathbf{L}^t(0,q)]^{-1}\mathbf{H}(0), \tag{11}$$

Notice that if $\mathbf{C}_{yjj}(0,q)$ is nonsingular for some $j \in \{1, 2, \ldots, r\}$, then (11) holds true even if $\mathbf{L}(k,q)$ and $\mathbf{L}(0,q)$ are replaced by $\mathbf{C}_{yjj}(k,q)$ and $\mathbf{C}_{yjj}(0,q)$, respectively. In case of $s < r$, we can solve (9) for $\mathbf{H}(k)$ and obtain, instead of (11),

$$\mathbf{H}(k) = \mathbf{L}(k,q)\mathbf{L}^t(0,q)[\mathbf{L}(0,q)\mathbf{L}^t(0,q)]^{\dagger}\mathbf{H}(0), \tag{12}$$

where † denotes the Penrose pseudo-inverse. Equation (11) or (12) is a simple closed form expression which relates the output cumulants with the MA coefficients. To utilize (11) or (12), we need $\mathbf{H}(0)$ of $\{\mathbf{H}(k)\}$.

To determine $\mathbf{H}(0)$, we consider (12) for $k = q$ and substitute $\mathbf{H}(q)$ into (6c) to obtain

$$\mathbf{H}(0)\mathrm{diag}[h_{i1}(0)h_{j1}(0), \ldots, h_{is}(0)h_{js}(0)]\mathbf{J}\mathbf{H}^t(0) = \mathbf{B}_{ij}, \quad i,j = 1, \ \ldots, r \tag{13a}$$

where

$$\mathbf{B}_{ij} = \mathbf{L}(0,q)\mathbf{L}^t(0,q)[\mathbf{L}(q,q)\mathbf{L}^t(0,q)]^{\dagger}\mathbf{C}_{yij}(q,0). \tag{13b}$$

Equations in (12) and (13b) are referred to as the *Cumulant-Based Identification (CBI)* equations. Note that (13a) is not generally guaranteed to have a solution when $\mathbf{B}_{ij}$ takes a different value from that given by (13b). However, (13a) always has a solution whenever $\mathbf{B}_{ij}$ is determined from (13b). We note that (13a) and (13b) are similar to (14a) and (14b) in the previous work [GI89].

We now discuss the issue of identifiability. Without any further assumptions, (13a) is not guaranteed to have a unique solution. However, as in the previous work [GI89], we can identify an equivalence class of MA(q) models that satisfy (5), and consequently (12) and (13a). In order to show this fact, we need the following notation and lemma.

Definition (Extended Permutation matrix).

A matrix $\mathbf{Q} \in R^{s\times s}$ is called an *extended permutation matrix* if it has a single nonzero entry which is equal to either -1 or 1 in each row and column, i.e., for i = 1,2, ...,s,

$$q_{ij} = 1 \text{ or } -1, \text{ if } j = j_i$$
$$\quad = 0 \quad , \text{if } j \neq j_i,$$

where $\{j_1, ..., j_s\}$ is a permutation of $\{1, 2, ..., s\}$.

Lemma. An s×s matrix $\mathbf{Q}$ is an extended permutation matrix if and only if

$$\mathbf{Q}\mathrm{diag}[q_{i1}q_{j1}, ..., q_{is}q_{js}]\mathbf{Q}^t = \mathbf{E}_i, \; i = j = 1, ..., s$$
$$= \mathbf{0}, \text{ otherwise,} \qquad (16)$$

where $\mathbf{E}_i = \mathrm{diag}[e_{11}, ..., e_{ss}]$ with $e_{ii} = 1$ and $e_{kk} = 0$ for $k \neq i$.

Using the above lemma, we obtain the following.

Theorem 1. Suppose *A1-A3* hold true. If $\{\mathbf{H}(k)\}_{k=1}^{q}$ satisfies (5), then $\{\tilde{\mathbf{H}}(k) \equiv \mathbf{H}(k)\mathbf{Q}\}_{k=1}^{q}$ also satisfies (5), where $\mathbf{Q} = \mathrm{diag}[\mathbf{Q}_1,\mathbf{Q}_2]$, $\mathbf{Q}_1 \in R^{\tau\times\tau}$, and $\mathbf{Q}_1 \in R^{(s-\tau)\times(s-\tau)}$ are extended permutation matrices. Conversely, if $\{\tilde{\mathbf{H}}(k)\}_{k=1}^{q}$ satisfies (5), then there exist extended permutation matrices $\mathbf{Q}_1 \in R^{\tau\times\tau}$, and $\mathbf{Q}_2 \in R^{(s-\tau)\times(s-\tau)}$ such that $\{\tilde{\mathbf{H}}(k) = \mathbf{H}(k)\mathrm{diag}[\mathbf{Q}_1,\mathbf{Q}_2]\}_{k=1}^{q}$.

Remark: Theorem 1 presents a result different from the previous work ([GI89b] [SW89]) in which the equivalence class is specified by the set of permutation matrices. On the other hand, in the fourth-order cumulant case, it is specified by the set of extended permutation matrices. The set of extended permutation matrix is larger than that of permutation matrices.

III. ESTIMATING MA MODELS BASED ON EIGEN-DECOMPOSITIONS

The basic motivation of our algorithm can be explained from a special case when $\mathbf{H}(0)$ is an orthogonal matrix. From (6c), it is readily seen that $\mathbf{H}(0)$ can obtained from the Schur decomposition [GO89] of $\mathbf{C}_{yii}(0,q)$ up to an extended permutation transformation provided $h_{i1}^2(q)$, $h_{i2}^2(q)$, ...,$h_{is}^2(q)$ are distinct for some i. Hence one of the important step is to find an orthogonalization transformation matrix $\mathbf{T}$ such that the column vectors of $\mathbf{TH}(0)$ are orthogonal. The standard orthogonalization techniques such as the Gram-Schmidt orthogonalization are not applicable because $\mathbf{H}(0)$ is unknown in this case. Such an orthogonalization transform matrix should be obtained only from the output process $\{\mathbf{y}(n)\}$. Here we consider the case when $\tau = s$. Putting

$$\mathbf{C} \equiv \sum_{i=1}^{r} \mathbf{C}_{yii}(0,q),$$

we obtain, from (6b),

$$\mathbf{C} = \mathbf{H}(0)\mathrm{diag}[d_1,\ldots,d_s]\mathbf{H}^t(0),$$

where, by *A3*,

$$d_i \equiv \sum_{j=1}^{r} h_{ji}^2(q) > 0, \text{ for } i = 1, \ldots, s.$$

Hence the singular value decomposition (SVD) of **C** must have the following form:

$$\mathbf{C} = \mathbf{U}\mathrm{diag}[\sigma_1^2, \ldots,\sigma_s^2, 0, \ldots, 0]\mathbf{U}^t$$

Let $\mathbf{U}_s$ be the submatrix of **U** consisting of the first s column vectors, it is then straightforward to show that

$$\mathbf{T} \equiv \mathrm{diag}[\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_s}]\mathbf{U}_s^t$$

will make the column vectors of **TH**(0) orthogonal. The algorithm for solving (6b) can be summarized as follows:

Eigen-Decomposition Based Algorithm

(I) Estimate the fourth-order cumulant matrices: $\{\mathbf{C}_{yij}(0,q), i,j = 1, \ldots, r\}$

(II) Compute

$$\mathbf{C} = \sum_{i=1}^{r} \mathbf{C}_{yii}(0,q)$$

and the singular value decomposition of **C**:

$$\mathbf{C} = \mathbf{U}\mathrm{diag}(\sigma_1^2, \ldots,\sigma_r^2)\mathbf{U}^t$$

where $\{\sigma_i^2\}$ are the singular values of matrix **C**.

(III) Estimate the number, s, of inputs from $\{\sigma_i^2\}$ and compute

$$\mathbf{T} = \mathrm{diag}(\sigma_1,\ldots,\sigma_s)^{-1}\mathbf{U}_s^t$$

where $\mathbf{U}_s$ consists of the first s column vectors of **U**.

(IV) Choose $\{\beta_{ij}\}$ such that $\mathbf{TC}^*\mathbf{T}^t$ has distinct eigenvalues where

$$\mathbf{C}^* = \sum_{i=1}^{r} \sum_{j=1}^{r} \beta_{ij}\mathbf{C}_{yij}(0,q)$$

(V) Compute the Schur decomposition of $\mathbf{TC}^*\mathbf{T}^t$, denoted by

$$\mathbf{TC}^*\mathbf{T}^t = \mathbf{Z}\Omega\mathbf{Z}$$

(VI) The estimation of **H**(0) up to right multiplication of a diagonal matrix Λ and an extended permutation matrix is given by

$$\hat{\mathbf{H}}(0) = \mathbf{T}^\dagger\mathbf{Z}$$

Theorem 2 Let $\hat{\mathbf{H}}(0)$ be obtained from the above algorithm. Then

$$\mathbf{H}(0)\mathbf{Q} = \hat{\mathbf{H}}(0)\Lambda$$

where $\mathbf{Q}$ is an extended permutation matrix and

$$\Lambda = \{(\hat{\mathbf{H}}^t(0)\hat{\mathbf{H}}(0))^{-1}\hat{\mathbf{H}}^t(0)\mathbf{B}\,\hat{\mathbf{H}}(0)(\hat{\mathbf{H}}^t(0)\hat{\mathbf{H}}(0))^{-1}\hat{\mathbf{D}}^{-1}\}^{1/4},$$

$$\mathbf{B} = \sum_{i=1}^{r}\sum_{j=1}^{r}\mathbf{B}_{ii},$$

$$\hat{\mathbf{D}} = \mathrm{diag}[\,\sum_{i=1}^{r}\hat{h}_{i1}^{2}(0),\ \ldots,\ \sum_{i=1}^{r}\hat{h}_{is}^{2}(0)].$$

Note that the matrix inside $\{.\}^{1/4}$ is diagonal with non-negative elements.

Remark: Although the above algorithm treats the cases when $\tau = s$, or equivalently, $\tau = 0$, the algorithm presented here is simpler than that in [TO90b].

IV. SIMULATION RESULTS

In order to examine the performance of our proposed method, a 3-input-6-output MA(2) model was chosen as follows:

$$\mathbf{H}(0) = \begin{bmatrix} 0.1493 & 0.2972 & 0.3319 \\ 0.0582 & 0.5304 & 0.2487 \\ 0.5370 & 0.3174 & 0.3930 \\ 0.0062 & 0.4030 & 0.4888 \\ 0.5721 & 0.1806 & 0.5043 \\ 0.5988 & 0.5785 & 0.4245 \end{bmatrix},\quad \mathbf{H}(1) = \begin{bmatrix} 0.7542 & 0.1123 & 0.1596 \\ 0.0382 & 0.6015 & 0.5762 \\ 0.4278 & 0.5303 & 0.1613 \\ 0.2638 & 0.3417 & 0.6041 \\ 0.2426 & 0.4758 & 0.0439 \\ 0.3438 & 0.3052 & 0.4996 \end{bmatrix},$$

$$\mathbf{H}(2) = \begin{bmatrix} 0.2135 & 0.4936 & 0.7627 \\ 0.6123 & 0.7311 & 0.1167 \\ 0.0942 & 0.0287 & 0.5870 \\ 0.2341 & 0.7045 & 0.0148 \\ 0.6235 & 0.1930 & 0.2172 \\ 0.3564 & 0.4219 & 0.1130 \end{bmatrix}.$$

Three mutually independent i.i.d. sequences were drawn randomly from a one-sided exponential distribution as the input signals. The sample variances of the three input signals were chosen to be unity. White Gaussian noise was added to each channel at variance σ^2. The Signal-to-Noise-Ratio (SNR) is consequently defined as

$$\mathrm{SNR} = -\,10\log_{10}\sigma^2$$

70,000 samples were used to estimate the fourth-order cumulant matrices. 300 Monte Carlo simulations were conducted in which $\mathbf{H}(0)$, $\mathbf{H}(1)$ and $\mathbf{H}(2)$ were estimated. The SNR was set at 20dB. Table 1, 2, and 3 show the sample mean, the standard deviation, and the root-mean-square-error of the estimates. The average of normalized root-mean-square-errors over all column vectors of $\hat{\mathbf{H}}(0)$, $\hat{\mathbf{H}}(1)$ and $\hat{\mathbf{H}}(2)$ was at the satisfactory level of 8.82%.

V. CONCLUSIONS

In this paper, we examine the identification of MA(q) models using the fourth-order cumulants. We found that an equivalence class of MA(q) models that can be identified by the fourth-order cumulants is slightly larger than that can be identified by the third-order cumulants. On the other hand, the computation involved in the identification using the fourth order cumulants is much simpler than that using the third-order cumulants. The simulation results show that the proposed method is computationally efficient.

REFERENCES

[BR75] D. R. Brillinger, *Time Series: Data Analysis and Theory*, New York: Holt, Rinehart and Winston, 1975.

[BR90] D. R. Brillinger, "A study of second and third-order spectral procedure and maximum likelihood in the identification of a bilinear system," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, pp. 1238-1245, 1990.

[CA89a] J. F. Cardoso,"Source separation using higher order moments," *Proc. IEEE ICASSP*, pp. 2109-2112 1989.

[CA89b] J. F. Cardoso, "Blind identification of independent components," *Proc. Workshop on Higher-Order Spectral Analysis*, pp. 157-160, 1989.

[CA90] J. F. Cardoso, "Eigen-structure of the fourth-order cumulant tensor with application to the blind source separation problem," *Proc. IEEE ICASSP*, pp. 2655-2658, 1990.

[CO89] P. Comon,"Separation on stochastic processes," *Proc. Workshop on Higher-Order Spectral Analysis*, pp. 174-179, 1989.

[GI89a] G. Giannakis, J. M. Mendel, "Identification of nonminimum phase systems using higher order statistics," *IEEE Trans Acoust., Speech, and Signal Processing*, vol. ASSP-37, No. 3, pp360-377.

[GI89b] G. Giannakis, Y. Inouye, and J. Mendel, "Cumulant based identification of multichannel moving-average models," *IEEE Trans. Automat. Contr.*, vol. AC-34, pp 783 - 787, July 1989.

[GI90] G. Giannakis, "On the identifiability of non-Gaussian ARMA models using cumulants," *IEEE Trans. Automat. Contr.*, vol. AC-35, pp.18-26, 1990.

[GO89] G. H. Golub and C. Van Loan, *Matrix Computations*. Second Edition. The Johns Hopkins University Press, 1989

[HA70] E. J. Hannan, *Multiple Time Series*. New York: Wiley, 1970.

[HA76] E. J. Hannan, "The identification and parameterization of ARMAX and state space forms, " *Econometrica*, pp. 713-723, 1976.

[IN84] Y. Inouye, "Maximum entropy spectral estimation for regular time series of degenerate rank," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 733-740, 1984.

[IN85] Y. Inoute,"Autoregressive model fitting for multichannel time series of degenerate rank: limit properties," *IEEE Trans. Circuit. Syst.*, vol. CAS-32, pp. 252-259, 1985.

[IN88] Y. Inouye, G. Giannakis, and J. Mendel, "Cumulant based parameter estimation of multichannel moving-average processes," *Proc. ICASSP*, pp. 1252-1255, April 1988.

[KE77] M. G. Kendall and A. Stuart, *The Advanced Theory of Statistics: Distribution Theory*, vol. 1, New York: Macmillan, 1977.

[LI90] K. S. Lii, "Identification and estimation of non-Gaussian ARMA processes," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, pp. 1266-1276, 1990.

[MA83] S. L. Marple Jr. and A. H. Nutall,"Experimental comparison of three multichannel linear prediction spectral estimators," *IEE Proc.* vol. 130, Part F, pp. 218-229, 1983.

[PH78] M. S. Phadke, and G. Kede, "Computation of the exact likelyhood function of multivariate MA models," *Biometrica*, pp. 511-520, 1978.

[RA85] M. R. Raghuveer and C. L. Nikias,"Bispectrum estimation: A parametric approach," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-33, pp. 1213-1230, 1985.

[RA86] M. R. Raghuveer, "Multichannel bispectrum estimation," *Proc. 3rd ASSP Workshop on Spectrum Estimation and Modeling*, pp. 21-24, 1986.

[RO67] E. A. Robinson, *Multichannel Time Sreries Analysis with Digital Computer Programs*, San Francisco: Holden Day, 1967.

[RO80] E. A. Robinson and S. Treitel, *Geophysical Signal Analysis*, New Jersey: Presentice-Hall, 1980.

[ROs85] M. Rosenblatt, *Stationary Sequences and Random Fields*, Boston: Birkhauser, 1985.

[RU89] P. Ruiz and J. L. Lacoume," Extraction of independent sources from correlated inputs: A solution based on cumulants," *Proc. Workshop on Higher-Order Spectral Analysis*, pp. 146-151, 1989.

[ST77] O. N. Strand, "Multichannel complex maximum entropy (autoregressive) spectral analysis," *IEEE Trans. Automat. Contr.*, vol. AC-22, pp.634-640, 1977.

[SW89] A. Swami, G. B. Giannakis and J. M. Mendel, "A unified approach to modeling multichannel ARMA processes," *Proc. ICASSP-89* pp. 2182-2185.

[TO89] L. Tong, V. C. Soon, Y. F. Huang and R. Liu, "Multiple source separation in noise," *Proc. 27th Annual Conf. on Commm. Contr. and Computing*, Urbana, IL, Oct. 1989.

[TO90a] L. Tong and R. Liu, "On solving simultaneous identification equations," Tech. Report, Dept. of Electrical Engineering, Univ. of Notre Dame, June 1990.

[TO90b] L. Tong, Y. Inouye and R. Liu,"A finite-step global convergence algorithm for the parameter estimation of multichannel moving average processes," submitted to *IEEE Trans. ASSP*.

Table 1 Sample means of the estimates

$\hat{H}(0)$	$\hat{H}(1)$	$\hat{H}(2)$
$\begin{bmatrix} 0.1441 & 0.2914 & 0.3440 \\ 0.0679 & 0.5313 & 0.2480 \\ 0.5413 & 0.3095 & 0.3998 \\ 0.0176 & 0.4044 & 0.4819 \\ 0.5744 & 0.1787 & 0.4978 \\ 0.5889 & 0.5831 & 0.4237 \end{bmatrix}$	$\begin{bmatrix} 0.7422 & 0.1338 & 0.1595 \\ 0.0647 & 0.5887 & 0.5872 \\ 0.4168 & 0.5396 & 0.1565 \\ 0.2668 & 0.3368 & 0.6002 \\ 0.2354 & 0.4772 & 0.0550 \\ 0.3531 & 0.0336 & 0.4882 \end{bmatrix}$	$\begin{bmatrix} 0.2139 & 0.4844 & 0.7608 \\ 0.6032 & 0.7286 & 0.1280 \\ 0.1087 & 0.0325 & 0.5696 \\ 0.2318 & 0.0845 & 0.0130 \\ 0.6117 & 0.2012 & 0.2059 \\ 0.3478 & 0.4290 & 0.0996 \end{bmatrix}$

Table 2 Standard deviations of the estimates

$\hat{H}(0)$	$\hat{H}(1)$	$\hat{H}(2)$
$\begin{bmatrix} 0.0358 & 0.0090 & 0.0106 \\ 0.0301 & 0.0141 & 0.0116 \\ 0.0090 & 0.0144 & 0.0083 \\ 0.0396 & 0.0144 & 0.0141 \\ 0.0132 & 0.0278 & 0.0132 \\ 0.0251 & 0.0050 & 0.0078 \end{bmatrix}$	$\begin{bmatrix} 0.0677 & 0.0196 & 0.0100 \\ 0.0518 & 0.0188 & 0.0167 \\ 0.0799 & 0.0062 & 0.0262 \\ 0.0282 & 0.0233 & 0.0080 \\ 0.0714 & 0.0093 & 0.0270 \\ 0.0195 & 0.0268 & 0.0421 \end{bmatrix}$	$\begin{bmatrix} 0.0378 & 0.0073 & 0.0060 \\ 0.0229 & 0.0069 & 0.0062 \\ 0.0173 & 0.0083 & 0.0053 \\ 0.0258 & 0.0072 & 0.0070 \\ 0.0339 & 0.0097 & 0.0068 \\ 0.0342 & 0.0053 & 0.0057 \end{bmatrix}$

Table 3 Root-mean-square-errors of the estimates

$\hat{H}(0)$	$\hat{H}(1)$	$\hat{H}(2)$
$\begin{bmatrix} 0.0362 & 0.0106 & 0.0161 \\ 0.0316 & 0.0142 & 0.0116 \\ 0.0099 & 0.0164 & 0.0107 \\ 0.0412 & 0.0144 & 0.0157 \\ 0.0134 & 0.0278 & 0.0148 \\ 0.0270 & 0.0068 & 0.0078 \end{bmatrix}$	$\begin{bmatrix} 0.0688 & 0.0291 & 0.0100 \\ 0.0582 & 0.0227 & 0.0200 \\ 0.0807 & 0.0112 & 0.0266 \\ 0.0284 & 0.0238 & 0.0089 \\ 0.0718 & 0.0094 & 0.0291 \\ 0.0216 & 0.0269 & 0.0437 \end{bmatrix}$	$\begin{bmatrix} 0.0416 & 0.0291 & 0.0562 \\ 0.0943 & 0.0260 & 0.0969 \\ 0.0281 & 0.0115 & 0.0489 \\ 0.0581 & 0.0115 & 0.0489 \\ 0.0825 & 0.0167 & 0.0619 \\ 0.0938 & 0.0082 & 0.0926 \end{bmatrix}$

Hangup in Decision-Directed Digital Carrier Synchronizers

Michael P. Fitz
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907
(317)-494-0592
email: mpfitz@whale.ecn.purdue.edu

William C. Lindsey
University of Southern California

Abstract

Digital carrier synchronization techniques have recently been the subject of a significant research effort due to the promise of fast, hangup-free acquisition. This paper demonstrates that this assertion is not strictly true; decision-directed phase estimation algorithms are inherently closed loop systems and have a hangup condition. This hangup condition exists with decision-directed processing even with "open-loop" filtering. This anomaly reduces the attractiveness of decision-directed phase estimations in fading channels.

I. Introduction

A significant amount of research [2,4,5,6,7,8,9,12] has recently investigated phase estimation structures which do not contain a phase-locked loop (PLL). These digital carrier synchronizers offer many performance and implementation advantages. The most significant implementation advantage of these architectures are their amenability to the use of gate array technology. The focus of this research is application to burst mode system architectures or the fading channel. Digital phase estimation was proposed as an alternative to the PLL for these applications since the hangup characteristic [3] of the PLL reduces its performance. These digital phase estimation structures were postulated to be free of hangup and hence are methods to provide near coherent performance in burst mode architectures or the fading channel. This conjecture is not strictly true; this paper will demonstrate that decision-directed (DD) digital phase estimators behave similarly to a PLL (i.e., they have a hangup condition and can "cycle" slip). These characteristics have not been exposed previously because the research on DD digital phase estimation has evaluated performance either via simulation or by assuming all decisions are correct. These analytical methods provide good engineering approximations to the performance but do not fully quantify the DD estimator performance.

In [12] a rigorous Markov analysis of a decision-directed (DD)-exponentially weighted (EW) phase estimation/demodulation architecture was presented. A Markov analysis revealed many subtleties in the operation of this demodulator. The most important subtlety recognized is that DD processing transforms the open loop architecture of the EW phase estimator into a closed loop architecture. This architecture is closed loop in the sense that the innovations for the carrier phase estimation scheme are a function of the phase error, $\phi(n)$, that is the difference between the reference phase angle $\hat{\theta}(n)$, and the carrier phase angle, θ_0. This operating characteristic is similar to the operating characteristics of a phase-locked loop. In fact it is the intention of this paper to demonstrate that an open loop estimator preceded by DD processing has a hangup condition (like the PLL). This anomaly must be considered when utilizing DD digital phase estimation schemes for burst mode or fading channel carrier synchronization.

This paper will be organized as follows. Section II will overview the DD phase estimation technique and present the analysis models. Section III presents the hangup anomaly of DD digital carrier synchronizers. A heuristic explanation of the causes of hangup and sample phase trajectories are presented. S-Curves, analogous to that of the PLL, are derived. The reverse nulls of these S-Curves confirm the existence of a hangup condition. Section IV presents the conclusions of this work.

II. System Models

This section reviews the models used for analysis in this paper. The channel considered is the simple additive white Gaussian noise (AWGN) channel with apriori known symbol timing. This model will be sufficient to demonstrate the hangup anomaly since it is a characteristic of the DD processing not the signal statistics or the method used to filter the innovations. The generic digitally implemented DD receiver is shown in Figure 1. To simplify the presentation, the analysis will be restricted to BPSK and QPSK modulations with pulse shapes which satisfy the Nyquist criterion for zero ISI. It can easily be shown [2] that the matched filter output in the AWGN channel with known symbol timing and negligible frequency uncertainty for BPSK modulation will have the form

$$x(n) = a_n \sqrt{E_b}\, \exp[j\,\theta_0] + \nu(n)\,. \tag{1}$$

Likewise the sample at the matched filter output for QPSK modulation can be represented as

$$x(n) = (a_n + jb_n)\sqrt{E_b}\, \exp[j\,\theta_0] + \nu(n)\,. \tag{2}$$

In both (1) and (2) the noise sequence, $\nu(n)$, is a zero mean, delta-correlated, discrete time, complex Gaussian random process with a variance of N_0, E_b is the bit energy, and θ_0 is the carrier phase. The modulation symbols, a_n and b_n, can take the values ± 1 with equal probability ($b_n = 0$ for BPSK). The conditional density function of x(n) (apropriately scaled) given θ_0 and the modulation symbols (a_n and b_n) is well known [1,10] in polar coordinates to be

$$\begin{gathered} p_x(A(n),\theta(n)\,|\,\theta_0,a_n,b_n) \\ = \frac{A(n)}{2\pi\sigma^2}\exp\left[-\frac{A(n)^2 - 2A(n)\cos\left[\theta(n)-\theta_0-\frac{\pi}{2}[1-a_n[1-\frac{b_n}{2}]]\right]+1}{2\sigma^2}\right] \\ -\pi \le \theta(n) \le \pi \\ 0 \le A(n) < \infty \end{gathered} \tag{3}$$

where $\sigma^2 = N_0/2E_s$, and E_s is the symbol energy[1]. Decision-directed processing attempts to remove the modulation from the matched filter output by multiplying this signal by the complex conjugate of the detected symbol (i.e. $\hat{a}_n - j\hat{b}_n$). If all decisions are correct the resulting signal, $\bar{x}(n)$, will be a complex sinusoid in AWGN. $\bar{x}(n)$ is the input to a phase estimator (open loop or a PLL) which derives the demodulator reference signal, r(n). In [12] an expression for the probability density function (pdf) of $\overline{x}(n)$ was derived for MPSK modulation. For BPSK modulation the pdf is expressed as[2]

---

[1] $E_s = E_b$ for BPSK and $E_s = 2E_b$ for QPSK

$$p_{\overline{x}}(\overline{A},\overline{\theta}\,|\,\hat{\theta}) = p_x(\overline{A},\overline{\theta}) + p_x(\overline{A},\overline{\theta}+\pi) \qquad \hat{\theta}-\frac{\pi}{2} \le \overline{\theta} \le \hat{\theta}+\frac{\pi}{2}$$

$$= 0 \quad \text{elsewhere} \tag{4}$$

where $\overline{A}$ and $\overline{\theta}$ are the amplitude and phase respectively of $\overline{x}(n)$. The expression for the pdf of $\overline{x}(n)$ for QPSK is

$$p_{\overline{x}}(\overline{A},\overline{\theta}\,|\,\hat{\theta}) = p_x(\overline{A},\overline{\theta}+\frac{\pi}{4}) + p_x(\overline{A},\overline{\theta}+\frac{3\pi}{4}) + p_x(\overline{A},\overline{\theta}-\frac{\pi}{4}) + p_x(\overline{A},\overline{\theta}-\frac{3\pi}{4})$$

$$\text{for } \hat{\theta}-\frac{\pi}{4} \le \overline{\theta} < \hat{\theta}+\frac{\pi}{4}$$

$$= 0 \quad \text{elsewhere.} \tag{5}$$

In [12] (4) and (5) were used to develop a statistical description of a DD reference signal. The DD-EW phase estimator was analyzed by the use of Markov chain theory and the Chapman Kolmogorov equation. For this paper (4) and (5) are used to expose and quantify the hangup anomaly in DD digital phase estimators.

III. Hangup in DD Phase Estimators

A. Introduction

Hangup is a phenomena characteristic of PLL based carrier phase estimation structures. Hangup is the condition when a PLL dwells for a prolonged time period at estimates which are greatly different from the actual carrier phase angle [3]. This dwell occurs when the phase error of the loop is in the vicinity of the phase detector (PD) reverse (or unstable) null. Even if the PD gain is made large in the region of the null (i.e. the sawtooth PD) equivocation around this null caused by noise will still produce a hangup condition. This phenomena is undesirable for modems which must acquire a phase estimate rapidly with high probability such as modems employed in time-division-multiple-access (TDMA) or slow frequency hopped (FH) systems. The poor performance of the PLL in the presence of signal fading has also been attributed to the hangup phenomena [8,9]. Open loop digitally implemented phase estimators were of interest to several authors [2,7,8,9] because of the elimination of the hangup condition. While it is true that open loop phase estimation of a sinusoid in additive noise does not exhibit a hangup condition, this is not always the case when the sinusoid is digitally modulated.

The hangup phenomena identified for DD open loop estimators for MPSK modulated signals occurs when the difference between the reference phase angle, $\hat{\theta}(n)$, and the carrier phase angle, θ_0 reside in specific regions. For BPSK modulation if the phase error, $\phi(n)$, resides near ±90° hangup can occur. Similarly for QPSK modulation hangup can occur if $\phi(n)$ is approximately ±45° or ±135°. Similar hangup conditions exist for higher order modulations. Considering (4) for BPSK and (5) for QPSK it is apparent that the phase of the innovation, $\overline{\theta}(n)$, input to the phase estimator will have had a restricted region in which it can take values. The center of this region is determined by the reference angle, $\hat{\theta}(n)$, and the size is determined by the modulation. For BPSK the span over which the phase angle of $\overline{x}(n)$ can take values is 180°, while for QPSK the span is 90°. Hangup can occur when the phase reference takes a value such that the decision boundaries determined by the phase reference are set approximately at the peak of the pdf of the matched filter output, x(n). Since the decision boundary splits the pdf of x(n), roughly half of the probability mass of $\overline{x}(n)$ will have a phase angle near one end of the defining interval and half will have a phase angle at the opposite end. This splitting of the probability mass is the cause of hangup in DD phase estimators.

---

[2] The dependence on n and the conditioning is dropped to simplify notation and the obvious modulo-2π phase reduction is assumed.

These characteristics of DD phase estimators can be further quantified by considering sample histograms of the various signals. Figure 2 and 3 demonstrate the splitting of the pdf of $\overline{x}(n)$ with example histograms for BPSK and QPSK respectively. The histograms in these figures are representative geometric mapping of the matched filter output, x(n), and DD processing output, $\overline{x}(n)$, onto an I and Q axis. Examination of Figures 2 and 3 reveals that though the phase error is large, the splitting of the pdf produces a situation where $E[\overline{\theta}(n)] = \hat{\theta}(n)$. This is a hangup condition and it is very similar to the hangup condition which plagues the demod-remod architecture [13] or equivalently the sawtooth PD [3,11] in a PLL. Examples of the hangup phenomena are seen in the phase trajectories in Figure 4 and 5.[1] These trajectories were produced by a DD-EW phase estimator [5,12] with $\beta = 0.875$. This estimation structure processes the quadrature amplitudes and is the simplest example of an "open loop" phase estimator. Both of the phase trajectories in the figures remain in the hangup region for over 60 symbols (note: 60 symbols is 4 time constants of the smoothing filter). The equivocation around the decision boundary by the matched filter output, x(n) is the cause of this hangup anomaly.

B. DD S-Curve

An S-curve in a closed loop synchronization architecture is defined as the equivalent PD characteristic [1]. This curve has traditionally been used for nonlinear analysis of synchronous control systems. The S-curve is a plot of the conditional expected value of the PD output given the value of the phase error and the received signal to noise ratio, i.e.

$$g(\phi) = E(\varepsilon(t) \mid \phi)$$

where $\varepsilon(t)$ is the error signal output from the PD. This curve is an important component for a nonlinear analysis of a PLL based synchronization system and provides insight into the performance characteristics of the system.

Though DD digital carrier synchronizers do not have an explicit PD component, an equivalent S-curve can be defined. This equivalent S-curve is defined as the conditional phase error after DD processing given the value of the phase error, i.e.

$$g(\phi) = E(\overline{\phi} \mid \phi) \tag{6}$$

where $\overline{\phi}$ is the innovation phase error after DD processing. This S-curve provides similar information as the S-curve of a PLL based synchronization system. Typically digital or open loop carrier synchronizers produce an unbiased phase estimate and this curve is a good indication of how the innovations will effect the estimator output. If a hangup condition exists a reverse null must be present in this S-curve. A reverse null indicates the average phase error of the innovation is zero and the phase reference will on the average remain the same. Hence developing the S-curves for the DD estimator will verify a hangup condition exists.

In developing this S-curve it can be assumed without loss of generality that the mean carrier phase angle, θ_0, is zero degrees. For BPSK modulation we also assume that the modulation is a sequence of all ones ($a_n = +1$ for all n). With these assumptions the PDF of the matched filter output sample phase for BPSK can be expressed by integrating over the amplitude in (3) as [10]

---

[1] In these two figures, data aided performance corresponds to the modulation being perfectly estimated. See [2] for more details.

$$p_\theta(\theta) = \frac{\exp\left[\frac{-E_b}{N_0}\right]}{2\pi} + \sqrt{\frac{E_b}{4\pi N_0}}\cos\theta\exp\left[\frac{-E_b}{N_0}\sin^2\theta\right]\left[1+\operatorname{erf}\left[\sqrt{\frac{E_b}{N_0}}\cos\theta\right]\right]$$

$$\text{for } -\pi \le \theta < \pi$$

$$= 0 \quad \text{otherwise} \tag{7}$$

where erf(x) is the Error Function defined as

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt\,.$$

By integrating over the amplitude in (4) BPSK DD processing produces a phase angle, $\bar{\theta}$, whose pdf can be expressed as

$$p_{\bar\theta}(\bar\theta) = p_\theta(\bar\theta) + p_\theta(\bar\theta + \pi) \quad \text{for } \hat\theta - \frac{\pi}{2} \le \bar\theta \le \hat\theta + \frac{\pi}{2}$$

$$= 0 \quad \text{elsewhere}\,. \tag{8}$$

The S-curve for a BPSK DD digital phase estimator can then be expressed using (3) in (6) as

$$g(\phi) = \hat\theta - \int_{\hat\theta - \frac{\pi}{2}}^{\hat\theta + \frac{\pi}{2}} \bar\theta[p_\theta(\bar\theta) + p_\theta(\bar\theta + \pi)]\,d\bar\theta$$

where $\bar\phi = \hat\theta - \bar\theta$. The resulting S-curve is seen in Figure 6. The null at +90° demonstrates that a hangup condition exists. This null condition indicates that even though the phase error is large, the average phase angle of the innovation will be that of the reference and hence the reference will on the average keep the same value. This is equivalent to the reverse null of a phase detector in a PLL which is the location at which a hangup condition exists.

A similar S-curve can be generated for DD processing in QPSK modulation. Again assuming $\theta_0 = 0°$ and an all one data sequence ($a_n = +1$, $b_n = +1$ for all n) the matched filter output sample's phase pdf can be expressed by again integrating over the amplitude in (3).

$$p_\theta(\theta) = \frac{\exp\left[\frac{-2E_b}{N_0}\right]}{2\pi} + \sqrt{\frac{E_b}{2\pi N_0}} \cos\left[\theta - \frac{\pi}{4}\right] \exp\left[\frac{-2E_b}{N_0} \sin^2\left[\theta - \frac{\pi}{4}\right]\right]$$

$$\left[1 + \mathrm{erf}\left[\sqrt{\frac{2E_b}{N_0}} \cos\left[\theta - \frac{\pi}{4}\right]\right]\right] \cdot$$

$$\text{for } -\pi \le \theta < \pi$$

$$= 0 \quad \text{elsewhere} \tag{9}$$

By integrating over the amplitude in (5) the pdf of $\bar{\theta}(n)$ for QPSK is expressed as

$$p_{\bar{\theta}}(\bar{\theta}) = p_\theta\left[\bar{\theta} + \frac{\pi}{4}\right] + p_\theta\left[\bar{\theta} + \frac{3\pi}{4}\right] + p_\theta\left[\bar{\theta} - \frac{\pi}{4}\right] + p_\theta\left[\bar{\theta} - \frac{3\pi}{4}\right]$$

$$\text{for } \hat{\theta} - \frac{\pi}{4} \le \bar{\theta} < \hat{\theta} + \frac{\pi}{4}$$

$$= 0 \quad \text{otherwise} \tag{10}$$

where $p_\theta(\ )$ is as defined in (9). The S-curve for a QPSK DD digital phase estimator can then by using (10) in (6) be expressed as

$$g(\phi) = \hat{\theta} - \int_{\hat{\theta}-\frac{\pi}{4}}^{\hat{\theta}+\frac{\pi}{4}} \bar{\theta} \left[p_\theta\left[\bar{\theta} + \frac{\pi}{4}\right] + p_\theta\left[\bar{\theta} + \frac{3\pi}{4}\right] + p_\theta\left[\bar{\theta} - \frac{\pi}{4}\right] + p_\theta\left[\bar{\theta} - \frac{3\pi}{4}\right]\right] d\bar{\theta}$$

This S-curve is seen in Figure 7. Again a null at +45° demonstrates that a hangup condition exists. Both the S-curves plotted in Figures 6 and 7 are partial plots. The S-curve is an odd function ($g(\phi) = -g(-\phi)$) and will be periodic (as are PLL S-curves) if the innovation phase error is defined as the difference between the innovation phase, $\bar{\theta}(n)$, and the closest stable lock point. For BPSK these stable lock points are θ_0 and $\theta_0 + 180^\circ$ while for QPSK they are θ_0, $\theta_0 + 90^\circ$, $\theta_0 + 180^\circ$, and $\theta_0 - 90^\circ$. The generation of the S-curve for DD digital phase estimators emphasizes the parallelism that exists between DD "open" loop estimators and traditional PLL architectures and proves analytically that a hangup condition exists. The fact that hangup exists in DD "open" loop estimators is a new result. The hangup phenomena should be carefully considered when utilizing a DD phase estimator in a burst mode architecture or on a fading channel.

IV. Conclusions

In this paper we have demonstrated that DD digital phase estimators have a hangup condition. Since the absence of hangup is the reason many researchers are advocating digital (open loop) carrier synchronizers [8,9] this is a significant result. Nonlinear (nondecision-aided) carrier synchronization (7) is recommended for the fading channel in lieu of decision-directed phase estimation.

References

[1] W. C. Lindsey and M. K. Simon, *Telecommunications Systems Engineering,* Prentice Hall, Englewood Cliffs, N.J., 1973.

[2] M. P. Fitz, "Open Loop Techniques for Carrier Synchronization," Ph.D. dissertation, Univ. of Southern California, Los Angeles, June 1989.

[3] F. M. Gardner, "Hangup in Phase-Lock Loops," *IEEE Trans. Commun.,* Vol. COM-25, pp. 1210-1214, Oct. 1977.

[4] P. Y. Kam, "Maximum Likelihood Carrier Phase Recovery for Linear Suppressed-Carrier Digital Data Modulations," *IEEE Trans. Commun.,* Vol. COM-34, pp. 522-527, June 1986.

[5] J. G. Proakis, P. R. Drouilhet Jr., and R. Price, "Performance of Coherent Detection Systems using Decision-Directed Channel Measurement," *IEEE Trans. Commun.,* Vol. CS-12, March 1964, pp. 54-63.

[6] P. Y. Kam, "Reception of PSK Signals Over Fading Channels Via Quadrature Amplitude Estimation," *IEEE Trans. Commun.,* Vol. COM-31.

[7] A. J. Viterbi and A. M. Viterbi, "Nonlinear Estimation of PSK Modulated Carrier Phase with Application to Burst Digital Transmission," *IEEE Trans. Inform. Theory,* Vol. IT-32, No. 4, July 1983, pp. 543-551.

[8] R. Haeb and H. Meyr, "A Systematic Approach to Carrier Recovery and Detection of Digitally Phase Modulated Signals on Fading Channels," *IEEE Trans. Commun.,* Vol. COM-37, July 1989, pp. 748-754.

[9] A. Aghamohammadi, H. Meyr, and Z. G. Ascheid, "Adaptive Synchronization and Channel Parameter Estimation Using an Extended Kalman Filter," *IEEE Trans. Commun.,* Vol. COM-37, November 1989, pp. 1212-1219.

[10] W. B. Davenport and W. L. Root, *An Introduction to the Theory of Random Signals and Noise,* McGraw-Hill Book Company, New York, NY, 1954.

[11] F. M. Gardner, *Phaselock Techniques,* John Wiley & Sons, 1979.

[12] M. P. Fitz and W. C. Lindsey, "Markov Analysis of a Decision-Directed Exponentially Weighted Phase Estimator and Demodulator." Submitted to *IEEE Trans. Commun.*

[13] C. L. Weber and W. K. Alem, "Demod-Remod Coherent Tracking Receiver for QPSK and SQPSK," *IEEE Trans. Commun.,* Vol. COM-28, December 1980.

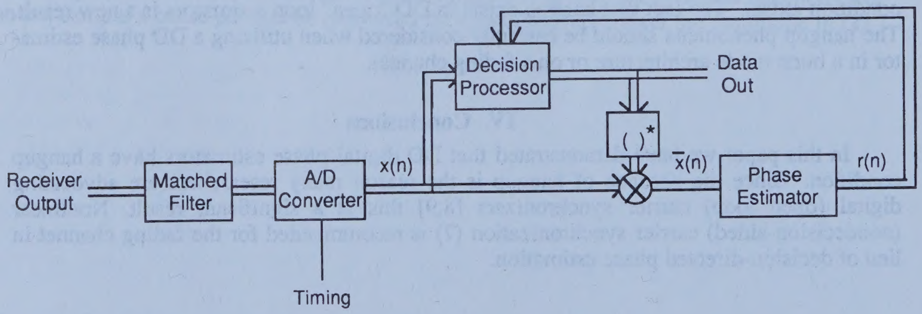


Figure 1. Block Diagram of a Decision-Directed Digitally Implemented Demodulator.

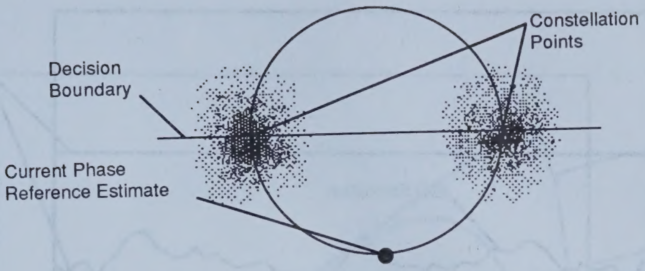

Constellation Points
Decision Boundary
Current Phase Reference Estimate

Figure 2 a). A Typical Histogram of the Matched Filter Output, x(n), for BPSK Modulation with a 90 ° Phase Reference Error.

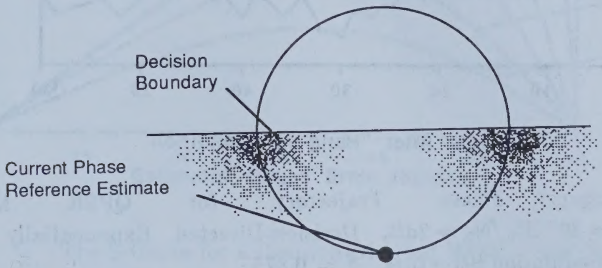

Decision Boundary
Current Phase Reference Estimate

Figure 2 b). The Resulting Histogram After Decision-Directed Processing, $\overline{x}(n)$, of the Signal Shown in a).

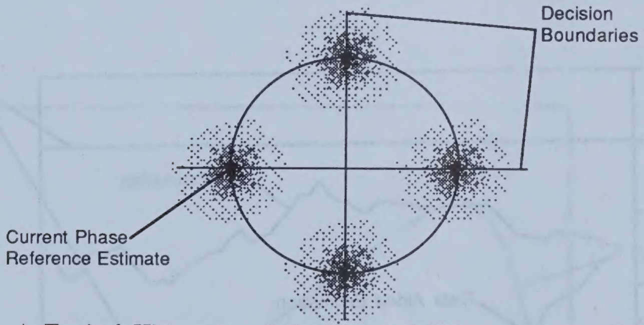

Decision Boundaries
Current Phase Reference Estimate

Figure 3 a). A Typical Histogram of the Matched Filter Output, x(n), for QPSK Modulation with a 45 ° Phase Reference Error.

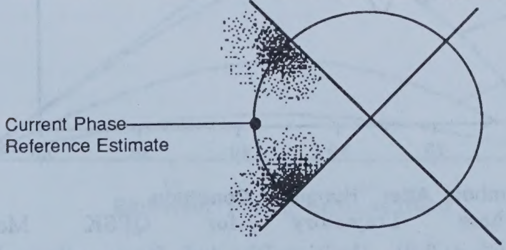

Current Phase Reference Estimate

Figure 3 b). The Resulting Histogram After Decision-Directed Processing, $\overline{x}(n)$, of the Signal Shown in a).

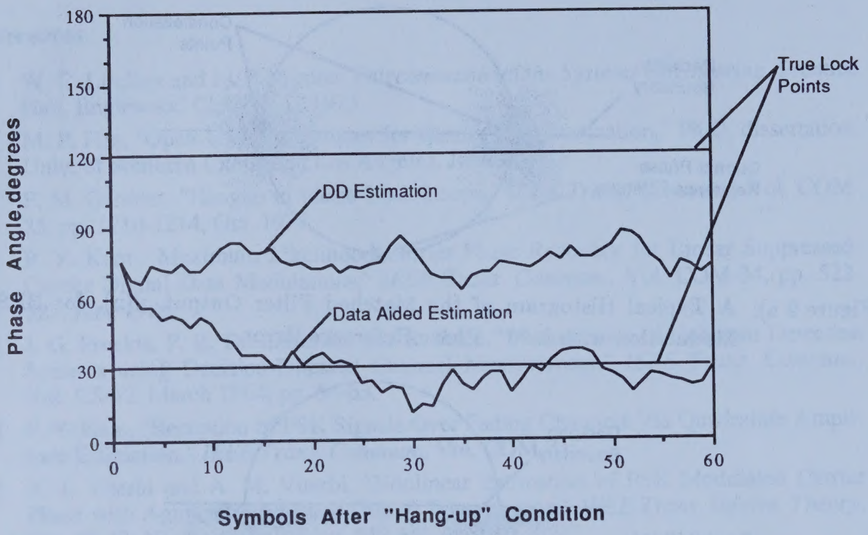


Figure 4. Hangup Phase Trajectory for QPSK Modulation. $\theta_0 = 30^\circ$, $E_b/N_0 = 2$dB, Decision-Directed Exponentially Weighted Demodulation Structure. $\beta = 0.875$

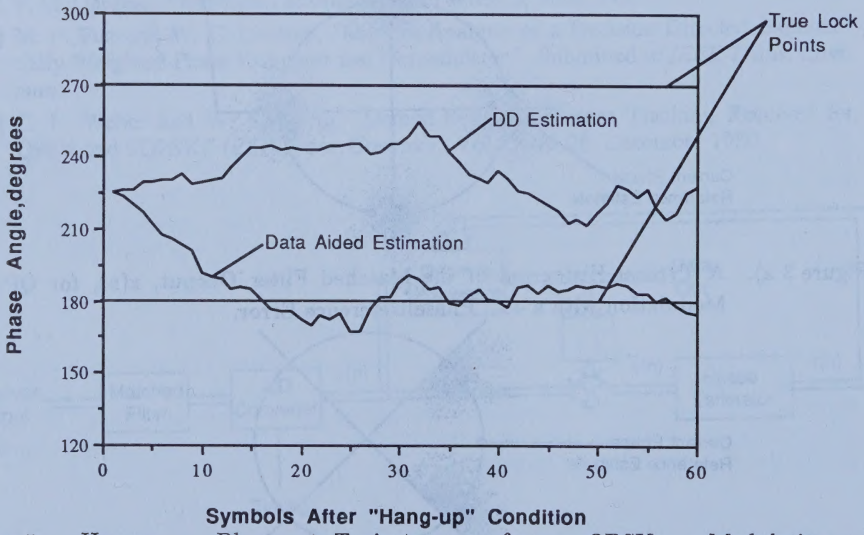


Figure 5. Hangup Phase Trajectory for QPSK Modulation. $\theta_0 = 180^\circ$, $E_b/N_0 = 2$dB, Decision-Directed Exponentially Weighted Demodulation Structure. $\beta = .875$.

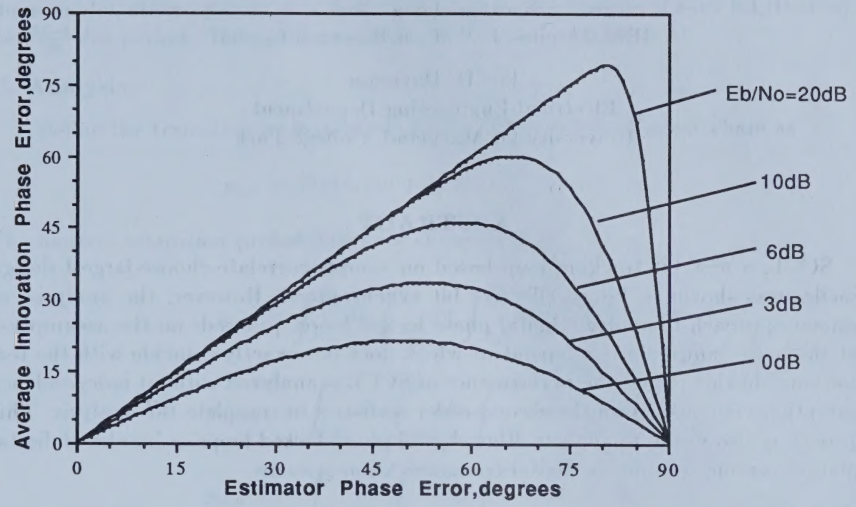


Figure 6. The S-Curve for a Decision-Directed BPSK Phase Estimator.

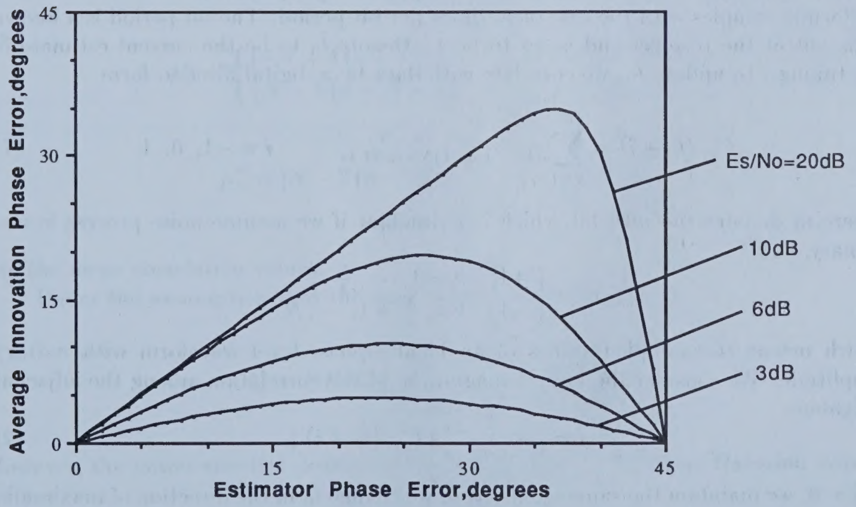


Figure 7. The S-Curve for a Decision-Directed QPSK Phase Estimator.

On The Loop Noise Bandwidth Of SCCL

Kwang-Cheng Chen
IBM Thomas J. Watson Research Center

Lee D. Davisson
Electrical Engineering Department
University Of Maryland, College Park

ABSTRACT

SCCL, a new bit tracking loop based on sample-correlate-choose-largest design princile, was shown to be an effective bit synchronizer. However, the analysis, as common approach to analyze digital phase locked loops, proceeds on the assumption that the noise samples are independent which does not exactly coincide with the real situations. In this paper, the performance of SCCL is analyzed without independence assumption via considering the second-order statistics to complete the analysis. This approach is also useful to analyze other digital phase-locked loops or baseband digital signal processing schemes in digital communication systems.

I. Introduction

SCCL, proposed recently by Chen and Davisson [1], is a new bit tracking loop closely to analogue the *MAP* optimal bit synchronizer with digital signal processing realization. The block diagram of SCCL is shown in the figure 1. The filtered baseband received waveform, $x(t)$, is sampled by an ideal A/D converter which means that each sample can be represented by infinite number of bits. The total filtering effect at the baseband is an ideal low pass filter with cutoff frequency B. Therefore, the net noise at the baseband is a band-limit Gaussian noise with zero mean and two-side spectral density $N_0/2$. A biphase-level signal set is applied in proposed systems. The sampler uniformly samples with the rate of N times per bit period. The bit period is a known constant at the receiver and is set to be 1. Denote $\hat{t}_0$ to be the current estimate of bit timing. To update $\hat{t}_0$, we correlate with data by a digital filter to form

$$C_m\left(\hat{t}_0 + i\right) = \sum_{k=1}^{N} g_k \cdot x_{(m-1)N+\hat{t}_0+k+i} \qquad i = -1,\ 0,\ 1 \tag{1}$$

where m denotes the mth bit which is redundant if we assume noise process is stationary, and

$$g_k = \begin{cases} +1, & k = 1, \cdots, \frac{N}{2} \\ -1, & k = \frac{N}{2}+1, \cdots, N \end{cases}$$

which means the sampled values of an ideal biphase-level waveform with unitary amplitude. We consider the largest magnitude of the correlation among the adjacent neighbors

$$j = \max_{i=-1,0,1}{}^{-1} \mid C_m\left(\hat{t}_0 + i\right) \mid \tag{2}$$

If $j = 0$, we maintain the same $\hat{t}_0$. If $j \neq 0$, we change $\hat{t}_0$ in the direction of maximum. That is, the estimate of timing in SCCL moves forward, backward, or unchanged, depending on these three correlation values.

Because $\{\hat{t}_0(m)\}$ forms a Markovian process, the behavior of the estimate of SCCL can be described by a Markov chain as shown in the figure 2. In this Markov chain model, state i means that $\hat{t}_0$ is closest to the ith sample of each bit, that is, at the $\frac{2i-1}{2N}$ bit period. The perfect synchronized state is state 1.

II. Analysis

Define the transition probabilities of the time invariant Markov chain as

$$p_{n,l} := Pr\{\hat{t}_0(m+1) = l \mid \hat{t}_0(m) = n\}$$

The nonzero transition probabilities are shown in [1] to be

$$p_{n,n} = \int_{-\infty}^{\infty} dy \int_{-|y|}^{|y|} dx \int_{-|y|}^{|y|} dz d_n(x,y,z)$$

$$p_{n,n+1} = \int_{-\infty}^{\infty} dz \int_{-|z|}^{|z|} dx \int_{-|z|}^{|z|} dy d_n(x,y,z)$$

$$p_{n,n-1} = \int_{-\infty}^{\infty} dx \int_{-|x|}^{|x|} dy \int_{-|x|}^{|x|} dz d_n(x,y,z) \tag{3}$$

where $d_n(x,y,z)$, $n = 1, \cdots, N$, is the joint Gaussian probability density function of $C_m(\hat{t}_0 + i)$, $i = -1, 0, 1$, with covariance matrix $\tilde{Q}$. (Thoughout this paper, $N+1 \equiv 1$ and $0 \equiv N$.)

$$d_n(x,y,z) = \frac{1}{4}(2\pi)^{-\frac{3}{2}} \mid \tilde{Q} \mid^{-\frac{1}{2}} \sum_{j=1}^{4} e^{-\frac{1}{2}\vec{\xi}^{n,j}\tilde{Q}^{-1}\vec{\xi}^{n,j\prime}}$$

where

$$\vec{\xi}^{n,j} = \left(x - \mu_j^{n-1}, y - \mu_j^n, z - \mu_j^{n+1}\right)$$

and

$$\mu_1^n = \begin{cases} [-N + 4(n-1)]A, & 1 \le n \le \frac{N}{2} + 1 \\ \left[N - 4\left(n - \frac{N}{2} - 1\right)\right]A, & \frac{N}{2} + 2 \le n \le N \end{cases}$$

$$\mu_2^n = -\mu_1^n$$

$$\mu_3^n = [N - 2(n-1)]A, \qquad 1 \le n \le N$$

$$\mu_4^n = -\mu_3^n$$

are the mean correlation values.

Under the assumption that the noise samples are independent,

$$\tilde{Q} = NBN_0 \begin{pmatrix} 1 & \frac{N-3}{N} & \frac{N-6}{N} \\ \frac{N-3}{N} & 1 & \frac{N-3}{N} \\ \frac{N-6}{N} & \frac{N-3}{N} & 1 \end{pmatrix}$$

However, the power spectral density of band-limited AGN (additive Gaussian noise) $n(t)$ is

$$S_n(f) = \begin{cases} \frac{N_0}{2}, & |f| \le B \\ 0, & \text{elsewhere} \end{cases} \tag{4}$$

Then, the noise samples may not be independent. Let us recall the stochastic version of Shannon's sampling theorem which was proved by Balakrishinan [3]. A continuous band-limited Gaussian process can be reconstructed in mean square sense by its uniform samples at the Nyquist rate. Furthermore, these Nyquist samples follow a Gaussian distribution $G(0, BN_0)$ independently, that is, they are i.i.d. [4].

In real systems, we sample N times per bit which is usually larger than the Nyquist rate, that is, $2B < \frac{1}{T}$, where T is the sampling period. The noise samples are not independent. The covariance matrix is different from what we previously got. Let

$$\tilde{Q} = \sigma^2 \begin{pmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{pmatrix} \tag{5}$$

be a representation of covariance matrix. To calcualte it, we need the following lemmas.

Lemma 1: Let $E[n_i n_j] = E[n_j n_i] = c_{|i-j|}$. Then,

$$c_{|i-j|} = (BN_0) \cdot Sinc(2BT|i-j|) \tag{6}$$

where

$$Sinc(x) = \frac{\sin \pi x}{\pi x}$$

Proof: Consider the frequency domain,

$$\begin{aligned} c_{|i-j|} &= E[n_i n_j] \\ &= E[n(iT)\, n(jT)] \\ &= R_n(|i-j|T) \\ &= \frac{1}{2\pi} \int_{-2\pi B}^{2\pi B} \frac{N_0}{2} e^{j\omega|i-j|T} d\omega \\ &= (6) \end{aligned}$$

As a matter of fact, this lemma can be generalized to any kind of low-pass filtering instead of the ideal low-pass filtering by considering the stochastic version of generalized sampling theorem.

Lemma 2: $\forall l$,

$$\begin{aligned} \sigma_N^2 &= E\left[\left(\sum_{k=1}^{N} n_{l+k}\right)^2\right] \\ &= NBN_0 + 2\sum_{i=1}^{N-1} (N-i)\, c_i \end{aligned} \tag{7}$$

A typical plot of σ_{16}^2 *v.s.* $1/2BT$ (sampling rate / loop noise bandwidth, oversampling ratio) is shown in the figure 3. At the Nyquist rate ($2BT = 1$), the total noise is $N(BN_0)$. When the sampling rate is higher than the Nyquist rate, the noise effect increases. The extra noise introduced from the sampling is named as *dependence* *noise* by the authors. When $\frac{1}{2BT} \to \infty$, the noise becomes highly dependent so that $\sigma_N^2 \to N^2 (BN_0)$.

Recall that the three estimates of SCCL for the mth bit with $\hat{l}_0(m) = n$ are

$$C_m(n-1) = \sum_{k=1}^{N/2} n_{(m-1)N+n+k-1} - \sum_{k=N/2+1}^{N} n_{(m-1)N+n+k-1}$$

$$C_m(n) = \sum_{k=1}^{N/2} n_{(m-1)N+n+k} - \sum_{k=N/2+1}^{N} n_{(m-1)N+n+k}$$

$$C_m(n+1) = \sum_{k=1}^{N/2} n_{(m-1)N+n+k+1} - \sum_{k=N/2+1}^{N} n_{(m-1)N+n+k+1}$$

where n_l's are identically distributed due to the stationary band-limited AGN assumption.

Proposition 3:

$$\sigma^2 = 2\sigma^2_{\frac{N}{2}} - \sum_{i=1}^{N/2}\sum_{j=1}^{N/2} c_{i+j-1} \tag{8}$$

Lemma 4: To evaluate ρ_1 in equation (5), we need:

(i)

$$\begin{aligned} \alpha_1 &= E\left[n_{(m-1)N+n} \left(\sum_{k=1}^{N/2-1} n_{(m-1)N+n+k} \right) \right] \\ &= E\left[\left(\sum_{k=1}^{N/2-1} n_{(m-1)N+n+k} \right) n_{(m-1)N+n+\frac{N}{2}} \right] \\ &= E\left[n_{(m-1)N+\frac{N}{2}} \left(\sum_{k=N/2+1}^{N-1} n_{(m-1)N+n+k} \right) \right] \\ &= E\left[\left(\sum_{k=N/2+1}^{N-1} n_{(m-1)N+n+k} \right) n_{(m-1)N+N} \right] \\ &= \sum_{k=1}^{N/2-1} c_k \end{aligned}$$

(ii)

$$\begin{aligned} \alpha_2 &= E\left[n_{(m-1)N+n} \left(\sum_{N/2+1}^{N-1} n_{(m-1)N+n+k} \right) \right] \\ &= E\left[\left(\sum_{k=1}^{N/2-1} n_{(m-1)N+n+k} \right) n_{(m-1)N+n+N} \right] \\ &= \sum_{k=N/2+1}^{N-1} c_k \end{aligned}$$

$$(iii) \qquad \alpha_3 = E\left[\left(\sum_{i=1}^{N/2-1} n_{(m-1)N+n+i}\right)\left(\sum_{j=N/2+1}^{N-1} n_{(m-1)N+n+j}\right)\right]$$

$$= \sum_{i=1}^{N/2-1}\sum_{j=1}^{N/2-1} c_{i+j}$$

Proposition 5:

$$\rho_1\sigma^2 = 2\sigma^2_{\frac{N}{2}-1} + 2\alpha_1 + 2c_{\frac{N}{2}} - 2\alpha_2 - c_N - 2\alpha_3 - \sigma_1^2 \tag{9}$$

where $\sigma_1^2 = E\left[n_l^2\right] = BN_0$.

Lemma 6: To find ρ_2, we need to evaluate:

$$(i) \quad \beta_1 = E\left[\left(n_{(m-1)N+n} + n_{(m-1)N+n+1}\right)\left(\sum_{k=1}^{N/2-2} n_{(m-1)N+n+k+1}\right)\right]$$

$$= E\left[\left(\sum_{k=1}^{N/2-2} n_{(m-1)N+n+k+1}\right)\left(n_{(m-1)N+n+\frac{N}{2}} + n_{(m-1)N+n+\frac{N}{2}+1}\right)\right]$$

$$= E\left[\left(n_{(m-1)N+n+\frac{N}{2}} + n_{(m-1)N+n+\frac{N}{2}+1}\right)\left(\sum_{k=N/2+1}^{N-2} n_{(m-1)N+n+k+1}\right)\right]$$

$$= E\left[\left(\sum_{k=N/2+1}^{N-2} n_{(m-1)N+n+k+1}\right)\left(n_{(m-1)N+n+N} + n_{(m-1)N+n+N+1}\right)\right]$$

$$= \sum_{i=1}^{2}\sum_{j=1}^{N/2-2} c_{i+j-1}$$

$$(ii) \quad \beta_2 = E\left[\left(n_{(m-1)N+n} + n_{(m-1)N+n+1}\right)\left(\sum_{k=N/2+1}^{N-2} n_{(m-1)N+n+k+1}\right)\right]$$

$$= E\left[\left(n_{(m-1)N+n+N} + n_{(m-1)N+n+N+1}\right)\left(\sum_{k=1}^{N/2-2} n_{(m-1)N+n+k+1}\right)\right]$$

$$= \sum_{i=1}^{2}\sum_{j=N/2+1}^{N-2} c_{i+j-1}$$

$$(iii) \qquad \beta_3 = E\left[\left(\sum_{k=1}^{N/2-2} n_{(m-1)N+n+k+1}\right)\left(\sum_{k=N/2+1}^{N-2} n_{(m-1)N+n+k+1}\right)\right]$$

$$= \sum_{i=1}^{N/2-2}\sum_{j-1}^{N/2-2} c_{i+j+1}$$

Proposition 7:

$$\rho_2\sigma^2 = 2\left(\beta_1 - \beta_2 - \beta_3 + \sigma^2_{\frac{N}{2}-2} - c_1 + 2c_{\frac{N}{2}} + c_{\frac{N}{2}+1} + c_{\frac{N}{2}-1} - c_N - \sigma_1^2\right) - c_{N+1} - c_{N-1} \tag{10}$$

Lemma 2 to Proposition 7 are based on straightforward algebraic manipulations and a stationary noise assumption. Details are neglected here. The plots of σ^2, $\rho_1\sigma^2$, and $\rho_2\sigma^2$ *v.s.* different oversampling ratios (sampling rate / loop noise bandwidth) are shown in the figure 4. Since a biphase-level signal set is applied, those noise covariances increase as the oversampling ratio increases; after reaching the maximum, the covariances decrease due to the fact that the digital samples become highly correlated and cancel each other because of g_k's. However, this does not imply that a high oversampling rate is desirable for SCCL. The reasons are

1. The resulting high sampling rate may not be achievable or practical.
2. The Markovian property of SCCL is destroyed by highly dependent samples. The condition to remain Markovian property valid is discussed later.

Applying the covariance matrix which takes the loop bandwidth into considerations, we can analyze the performance of SCCL by the same analytical methods discussed in [1]. The transition matrix is calcualted first from (3). The steady-state probability distribution, $(\pi_1, \cdots, \pi_N)$, can be solved by

$$\vec{\pi}P = \vec{\pi} \tag{11a}$$

$$\sum_{n=1}^{N} \pi_n = 1 \tag{11b}$$

The steady-state mean suqare error with fixed θ is

$$\sigma_\theta^2 = \sum_{n=1}^{N/2} \pi_n \left(\frac{n-1+\theta}{N}\right)^2 + \sum_{n=N/2+1}^{N} \pi_n \left(1 - \frac{n-1+\theta}{N}\right)^2 \tag{12}$$

where θ means a fraction of sampling period which is an inherent phase jitter in baseband digital phase locked loops or bit synchronizers. The numerical results for $\sigma^2_{\frac{1}{2}}$ with different $2BT$'s are shown in the figure 5. The mean acquisition time which means the expected time for the estimate first into the perfect synchronized state (state 1) is calculated as follows. First solve a set of linear equations

$$F_1 = 0$$

$$F_n = p_{n,n-1}F_{n-1} + p_{n,n}F_n + p_{n,n+1}F_{n+1} + 1 \qquad n = 2, \cdots, N \tag{13a}$$

Suppose that the epoch initially falls into each state with equal probability. The mean acquisition time, F, is

$$F = \frac{1}{N}\sum_{n=1}^{N} F_n \tag{13b}$$

The numerical results are depicted in the figure 6. The curves of $2BT = 0.13333$ in figure 5 and 6 correspond to the maximum noise covariance while the Markov chain model still reasonably holds.

An important consideration is whether the first-order Markov chain model is suitable for the analysis of dependent noise samples. With the aid of Lemma 2, we may have a rule of thumb to judge this consideration.

Proposition 8: *The Upper Bound Of Correlation Between Two Noise Samples*

$$\begin{aligned} |c_k| &= \left|(BN_0)\,\frac{\sin 2BT\pi k}{2BT\pi k}\right| \\ &\leq (BN_0)\,\frac{1}{2BT\pi k} \end{aligned} \tag{14}$$

where $k \geq 1$.

The closest samples while deciding the timing estimates during the $(m-1)$th and mth bit periods have correlation c_{N-1}. In order to keep our first-order Markov chain model to represent the behavior of SCCL closely, C_{N-1} should be small enough so that our analytical results are reasonable.

Rule Of Thumb For The Validity Of First-Order Markov Chain Model

If $2BT \geq \frac{10}{\pi(N-1)}$, *the first-order Markov chain model is approximately valid.*

The above conclusion is reached by assuming that $C_{N-1} \leq \frac{1}{10}BN_0$ results in a small correlation so that the behavior of estimate of SCCL is close to a first-order Markov process.

III. Conclusions

This paper provides an approach to deal with dependent noise samples by only considering their statistics. The exact method to analyze this problem involves the sample-path problem of filtering a white Gaussian process. The authors have not known any simple conclusion through such an approach until this moment.

Although it is difficult to define the loop bandwidth of any digital phase locked loop, especially for SCCL, this paper gives us another way to understand how loop noise bandwidth affects the loop performance. Most importantly, it suggests us that many digital communication systems applying digital signal processing techniques can be more practically analyzed by similar approach.

Finally, a biphase-level signal set is more desirable in high sampling rate systems than an NRZ signal set because of its tolerance to dependent noise in addition to other well known reasons such as a zero crossing for each bit.

References:

[1] K.C. Chen, L.D. Davisson, "Analysis Of A New Bit Tracking Loop — SCCL", submitted for publications, also partially presented at the International Symposium on Information Theory, Kobe, Japan, 1988.

[2] J. Wozencraft, I. Jacob, *Principles Of Communication Engineering*, 1967.

[3] A.V. Balakrishinan, "A Note On The Sampling Principle For Continuous Signals", *IRE Tr. on Information Theory*, Jan. 1957, pp.143-146.

[4] B.P. Lathi, *Random Signals And Communication Theory*, 1968.

[5] A. Papoulis, *Signal Analysis*, 1977.

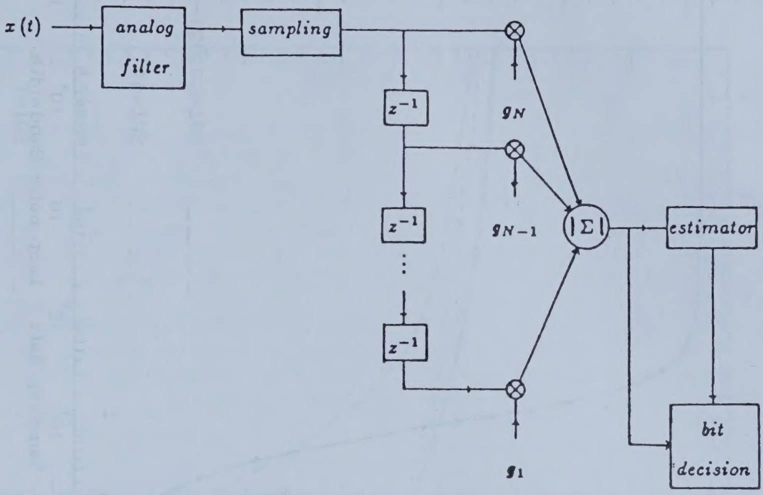

x(t)
analog filter
sampling
z^{-1}
z^{-1}
z^{-1}
g_N
g_{N-1}
g_1
$|\Sigma|$
estimator
bit decision

Figure 1 Block Diagram Of SCCL

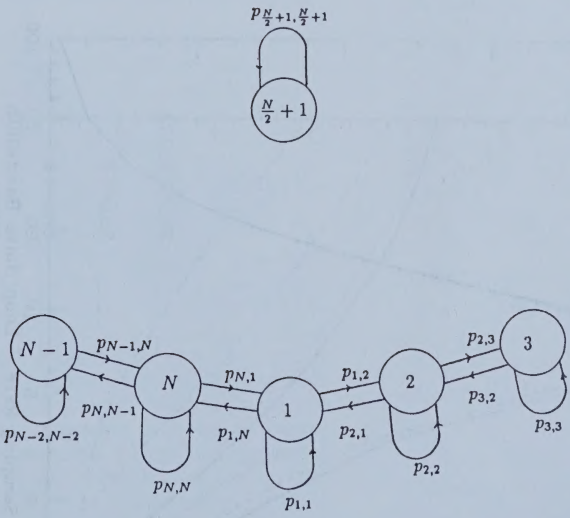

$p_{\frac{N}{2}+1,\frac{N}{2}+1}$
$\frac{N}{2}+1$
$N-1$
$p_{N-1,N}$
N
$p_{N,1}$
1
$p_{1,2}$
2
$p_{2,3}$
3
$p_{N,N-1}$
$p_{N-2,N-2}$
$p_{1,N}$
$p_{2,1}$
$p_{3,2}$
$p_{3,3}$
$p_{N,N}$
$p_{1,1}$
$p_{2,2}$

Figure 2 Markov Chain Model

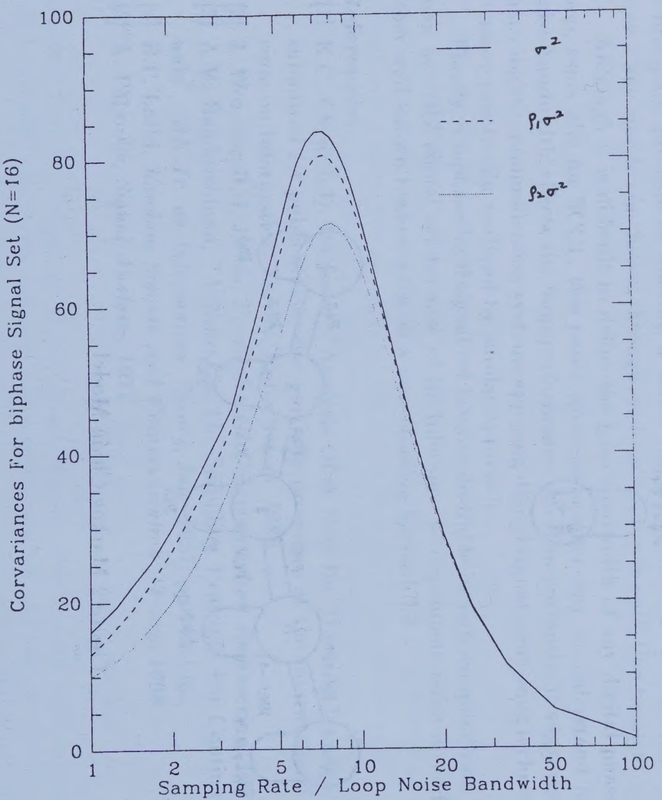


Figure 4 Noise Covariances *v.s.* Oversampling Ratio

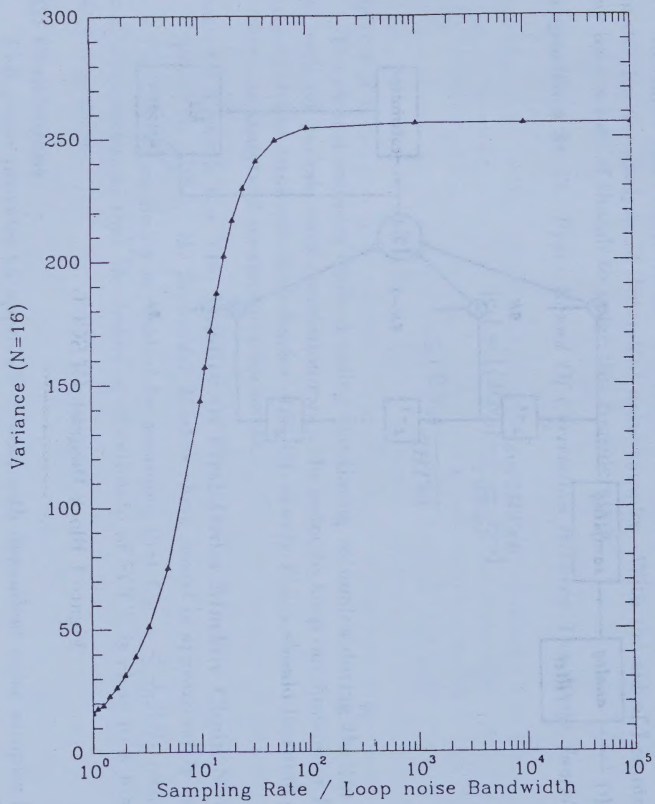


Figure 3 σ_{16}^2 *v.s.* Oversampling Ratio

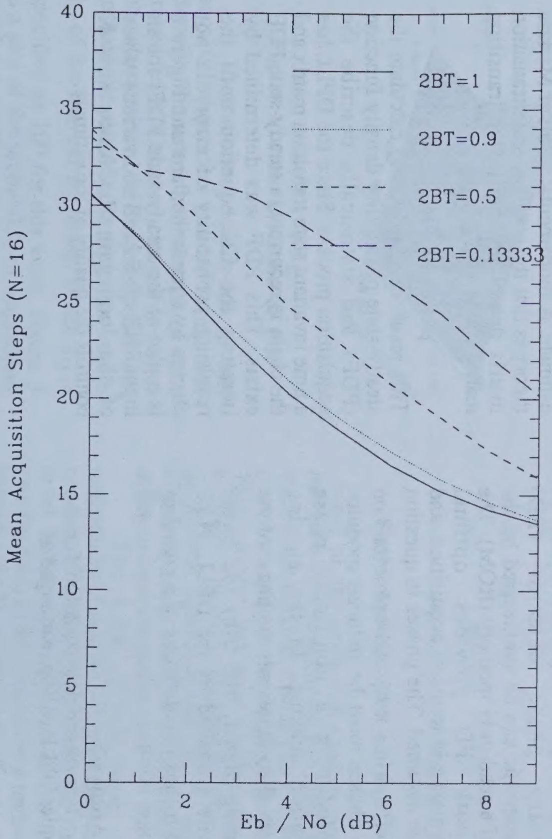


Figure 6 Mean Acquisition Time *v.s.* $2BT$

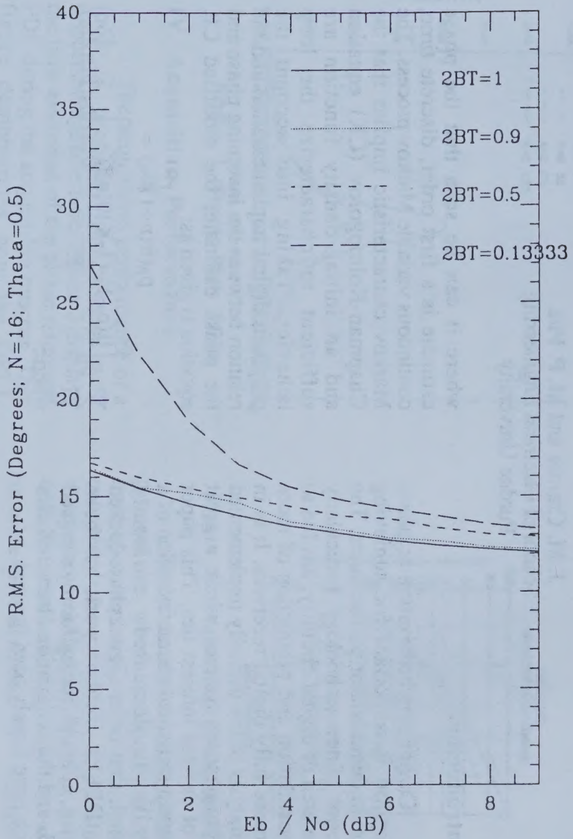


Figure 5 R.M.S. Error *v.s.* $2BT$

The Statistical Characterization of the Acquisition Process in a First Order Digital Phase-Locked Loop.

J. M. Cramer and M. P. Fitz
School of Electrical Engineering
Purdue University

I. Introduction

Carrier synchronization or phase estimation is important for achieving optimum demodulator performance. The evolution of new technology, particularly in the area of digital circuitry, has led to the postulation and production of more and more fully digital receivers. It is in the analysis of the digitally implemented synchronization systems where a great deal of current interest lies. This paper presents a statistical characterization of a digitally implemented, uniformly sampled, first-order digital phase-locked loop (DPLL). Similar work has been reported, although the uniformly sampled DPLL and the acquisition characteristics of such have largely been ignored.

II. Assumptions

Assumption made to facilitate the analysis are that the noise is additive white Gaussian (AWGN) and that the frequency offset is zero. Due to the proposed digital implementation, an arbitrary phase detector (PD) characteristic can be synthesized by the use of a read-only memory (ROM). The sawtooth PD provides optimal performance in terms of acquisition, and will be assumed. The phases in question are defined in $[-\pi,\pi]$. All operations on these phases must be reduced modulo-2π, producing a non-linear phase estimation scheme. In the ensuing analysis, $\theta(n)$ represents the phase of the incoming signal, and $\hat{\theta}(n)$ the phase estimate produced by the DPLL. Note that $\theta(n)$ has the statistics of a signal in bandpass noise.

III. Analysis

The difference equation for the first order DPLL can be rearranged as

$$\hat{\theta}(n+1) = K\theta(n) + (1-K)\,\hat{\theta}(n),$$

where it can be seen that the phase estimate is a first order, discrete time, continuous variable Markov process. The Markov characteristic implies that the Chapman-Kolmogorov (CK) equation and an initial density function are sufficient to characterize the loop behavior. Taking into account the proposed digital implementation and the relation between the incoming phase and the phase estimate, the modified CK equation is given as:

$$p_\theta(\hat{\theta}(n+1)|\hat{\theta}_o) = \sum_{\Omega} p_\theta\left[\frac{\hat{\theta}(n+1)-(1-K)\hat{\theta}(n)}{K}\middle|\hat{\theta}(n)\right] p_n(\hat{\theta}(n)|\hat{\theta}_o)$$

where:

$p_n(\cdot)$ = PDF of phase estimate at time n.

$p_\theta(\cdot)$ = PDF of phase of incoming signal.

From this equation, a state transition matrix can be determined for various values of the current and next phase estimate. A fundamental result of Markov theory is that the one-step state transition matrix determines the k-step transition matrix via

$$p(k) = p\left(\hat{\theta}(n+1)|\hat{\theta}(n)\right)^k.$$

This result can be used to calculate the steady-state probability density function (PDF) and statistically describe the acquisition process. Since the DPLL has a time invariant state transition matrix and the chain is recurrent, a steady-state PDF exists. This PDF was determined by iterating the CK equation until the resulting probability of error did not change by a prescribed amount. Figure 1 is a plot of the steady-state PDFs for an input SNR of -5.22dB and various values of the loop gain K. Notice how K controls the quality of the estimate.

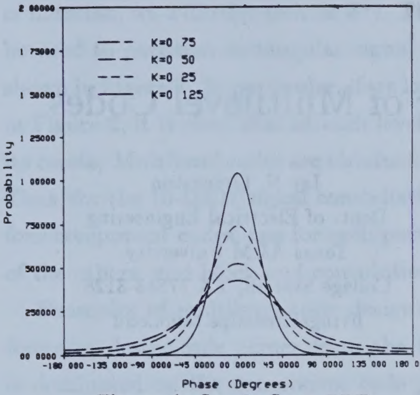


Figure 1. Steady-State PDFs

IV. Acquisition Performance

The acquisition performance of a loop is probably the single most important characteristic. Figure 2 displays the time evolution of the phase reference PDF during the acquisition process. This figure demonstrates how the DPLL iteratively moves the probability mass from a uniform distribution to one concentrated around the actual carrier phase.

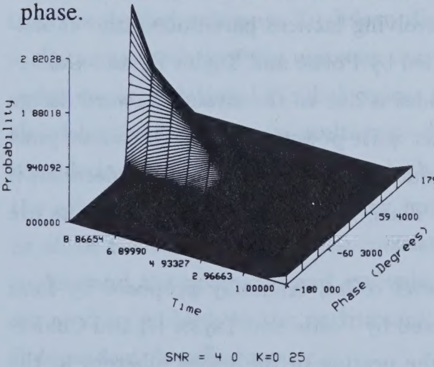


Figure 2. Time Evolution of PDF

A statistical description of this acquisition process can be formulated through the use of an absorbing state imbedded in a Markov chain. For carrier phase acquisition, the absorbing state will include the phase states which lie within ϕ_o of the true carrier phase, where ϕ_o is specified as 10^o for this work. Figure 3 is a plot of the probability of acquisition as a function of time for several values of K.

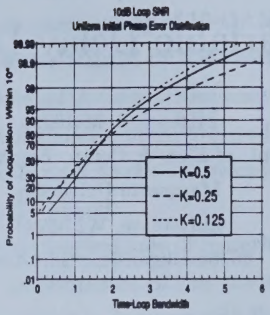


Figure 3. Acquisition Characteristics

Two functions of the acquisition process are worth noting. First is the hangup phenomenon of the DPLL, caused by the equivocation of probability mass about the phase detector reverse null. This can be seen in Figure 4, where an initial phase error of 180^o was used.

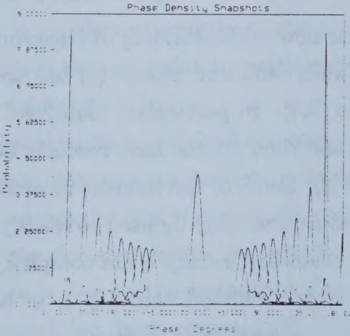


Figure 4. Acquisition of Impulse

Most of the probability mass moves toward a zero phase error, but after each step except the first, a small piece of mass moves away from the equilibrium point. This process has been designated as hangup. The second point is that the acquisition characteristics, as shown in Figure 3, are a function of K, which is different from the analog case.

The work in this paper has statistically characterized the acquisition process of a uniformly sampled first-order DPLL. The selection of the loop gain has been determined to be a trade-off between rapid acquisition (K->1) and estimate fidelity (K-> 0).

Variable Delay Decoding of Multilevel Codes

Michael G. Mulligan
Dept. of Elec. & Comp. Engineering
Marquette University
Milwaukee, WI 53233
6636mulligan@mucsd.bitnet

Jay N. Livingston
Dept. of Electrical Engineering
Texas A&M University
College Station, TX 77843-3128
livingst@eemips.tamu.edu

Abstract

The trade-off between decoding delay and performance of multilevel codes is considered in this paper. When using convolutional codes as component codes in a multilevel coding scheme, the delay of a multistage decoder can be quite substantial. Modifications can be made to the multistage decoder which reduce this delay, although a performance penalty is incurred.

The new understanding of Ungerboeck's "mapping by set partitioning" [8] provided by Calderbank and Sloane [1] has sparked interest in codes based upon lattices and cosets [3,4]. In particular, multilevel codes involving lattices partitioned into sublattices and their cosets, have recently been studied by Pottie and Taylor [7] and Calderbank [2]. Some of the interest in multilevel codes is due to the straightforward design procedure that may be used to readily construct quite powerful codes. Of course, powerful codes are usually quite complex, which places a heavy burden on the decoder, if maximum-likelihood decoding is to be used. Thus, reduced-complexity decoders are often considered in such situations.

Multistage decoding procedures for multilevel codes, originally proposed by Imai and Hirakawa [6], have recently been reconsidered by Pottie and Taylor [7] and Calderbank [2]. These procedures are based upon the nesting of the codes inherent in the partitioning of the signal constellation. The component codes in a multilevel code are thus decoded individually, with the code which operates on the first partition being decoded first. The outputs of this first decoder, along with the delayed channel symbols, become the inputs to the decoder for the code of the second partition, and so on. The simplicity of this approach is obtained at the cost of reduced performance and increased delay. In this paper we consider ways to reduce the delay of multistage decoding structures, and the effect of reduced delay on code performance.

In this paper we will concentrate on rectangular signal constellations defined by subsets of the lattice Z^2. (Actually, we will work with $Z^2+(1/2,1/2)$, but for simplicity of notation, we will refer to it as Z^2). The partition chain $Z^2/RZ^2/2Z^2/2RZ^2/\ldots$ may be used to partition rectangular signal constellations such as 4-, 16-, and 64-QAM as shown in Figure 1. In particular, if we look at the binary partitioning of 16-QAM shown in Figure 2, it is clear that at each level the points are determined by a sublattice and its cosets. Multilevel codes are obtained by encoding each of these partitions separately. Thus, for the 16-QAM signal constellation, we could construct a multilevel code out of four component codes, one for each partition. Each code may be chosen independently of the others, and block and convolutional codes may be mixed.

Examples of multilevel code design are provided by Calderbank [2]. Perhaps the foremost design rule comes from the fact that the performance of the overall code is dominated by the component code with the lowest minimum (squared) Euclidean distance between codewords or code sequences. With the knowledge that the Euclidean distance of each code is simply the Hamming distance of the code multiplied by the intrasubset distance at that level of the partition chain, code design becomes an easy task. Note, however, that the performance is not fully determined by this one parameter (the minimum distance), especially when suboptimal decoders are employed. Pottie and Taylor [7] point out that staged decoding increases the effective path multiplicities, and Calderbank [2] enumerates the multiplicities for a number of multilevel coset codes. Still, from a design point-of-view, the first step is to assign codes to partitions such that the desired minimum Euclidean distance is achieved.

In our research we focus our attention on multilevel codes in which all the component codes are convolutional (trellis) codes. Once again, our primary concern is to reduce the delay incurred by the multistage decoding approach, and therefore we start by considering parallel decoding, in which each of the decoders works independently of the others. Clearly, the only delay incurred here is due to the individual decoders, as all of the excess delay of the multistage approach has been removed. In order to understand the operation and evaluate the performance of such a decoding strategy we need to go back to the partitioning of the signal constellation and consider what independent decoding means.

For example, working with 16-QAM we will have four separate encoders/decoders for the four partitions of Figure 2. The question at hand is: With no information from the other decoders, how does each decoder operate? To answer this, we must consider that for any one of the encoders, each output bit selects one of two subsets of the original constellation, although at each level the constellation is divided differently. To illustrate this, we refer to Figure 3. Here we have at each level gathered the points from Figure 2 corresponding to an encoder output of zero, and separated them from those corresponding to an encoder output of one.

In the absence of information from the other decoders then, each decoder is faced with eight possible symbols for the zero bit, and eight for the one bit. Each decoder must take this into account in the calculation of branch metrics, computing distances to the *closest* symbol from the zero subset and from the one subset. We refer to these distances as subset metrics, which may be calculated in a number of ways. Perhaps the simplest approach conceptually is that of computing the distance from the received symbol to each of the symbols in the subset, and choosing the smallest such distance as the subset metric. However this potentially involves a great deal of computation, and in the name of simplicity we reject this approach.

Another way, which is to be preferred, is to map each of the subset members onto a representative point and then to apply the same mapping to the received signal point. In this way only one distance needs to be computed, and the subset metric is obtained directly. Note however, that because at each level the subsets are not the same, the required mappings will vary depending upon the partition level. In most cases, the mappings will involve some combination of rotations, translations, foldings, and/or modulo arithmetic.

This form of parallel decoding we refer to as "subset decoding" because just as the outputs of each encoder are used to select a particular subset, so each decoder operates by trying to determine which subset was selected. Consider again Figure 3, in which the four separate levels are labeled to correspond with four codes, $C1$, $C2$, $C3$, and $C4$. Associated with each of the sixteen possible combinations of outputs from the four encoders there is a unique set of subsets. For any such combination of outputs, the intersection of these subsets will consist of only a single point, the transmitted signal. Our decoding approach is essentially the inverse of this procedure: We take the received signal and try to determine each of the four subsets whose intersection produced the transmitted signal.

Although the simplicity of this approach affords a reduction in both delay and complexity, the performance will clearly be inferior to the multistage approach, since there is no information passed between decoders. In fact, looking at Figure 3 we see that the intrasubset distance at each level is the distance between points in the original constellation. Therefore we expect the minimum distance to be effectively reduced by such an approach. This is illustrated by the following examples.

For simplicity, we restrict these examples to two-level codes based upon the two-level partition of 4-QAM, employing both two-state and four-state rate 1/2 convolutional codes. The intrasubset distances, for this constellation normalized to unit bit energy, are $(4, 8, \infty)$ for the three subset levels. The codes used are the maximum Hamming distance codes, defined by the generators [3 2] for the two-state code, and [7 5] for the four-state code. The Hamming distance of the two-state code is 3, while for the four-state code it is 5. Using a rate 1/2 code on each partition gives us an overall code rate of 2/4. A fair comparison to these coded schemes would be uncoded binary PSK, which has a normalized Euclidean distance of 4, but taking into account the code rate, we need to double this uncoded distance to 8. We consider two cases: both partitions encoded with two-state codes, and the first partition encoded with the four-state code while the second partition is encoded with the two-state code.

In the first case we calculate Euclidean distances of 12 and 24 for the first and second codes, respectively. Clearly, the performance will be dominated by the first code, and we expect an improvement of approximately 1.8 dB over uncoded PSK at high signal-to-noise ratios. From the simulation results presented in Figure 4, we see that for maximum-likelihood decoding this improvement is indeed obtained. (Note that the simulation results have been fit to a curve defined by a single complementary error function. Extrapolation beyond the range of experimental results is risky at best.) We also show on this plot the simulation results for the case of parallel decoding. In this case, even though the effective Euclidean distance of the second-level code is reduced to 12, this is not less than the Euclidean distance of the first-level code, and thus the roughly 1 dB loss in performance can be attributed to the sub-optimality of the decoding procedure.

The second example illustrates a different situation. Using the four-state code on the first partition we obtain a Euclidean distance of 20, while the two-state code operating on the second partition yields a Euclidean distance of 24. Once again, the performance of the maximum-likelihood decoder is dominated by the distance of the first code, and we expect in this case a 4 dB improvement over uncoded PSK. The simulation results in Figure 5 show almost a 4 dB improvement, confirming our expectations. For parallel decoding we expect the Euclidean distance of the second code to be reduced to 12, thereby reducing the coding gain to the 1.8 dB of the first example. Although on first glance our simulation results might seem to verify this, the coding gain in this situation is more properly viewed as an average of the two coding gains, 1.8 dB and 4 dB, less the penalty for the sub-optimal decoding approach, and not simply the lesser of the two.

Both of these examples emphasize the fact that parallel decoding can incur a substantial performance loss in comparison to maximum-likelihood decoding, although the trade-off is, of course, for reduced complexity. In presenting these two approaches we have essentially considered the two most extreme cases. In-between lie a number of approaches with more delay than parallel decoding and less complexity than maximum-likelihood decoding. We consider essentially only one type of intermediate approach, the multistage approach advocated in [2,6,7], but with an eye to minimizing the amount of delay incurred.

Using the multistage approach, we consider only decoders in which the heirarchy of codes is preserved. That is, the first decoder will pass information along to the second and all subsequent decoders, but must itself operate independently, while the second decoder passes information on to the third and higher decoders, while it receives information only from the first decoder, and so on. The refinements to this approach may be placed into one of two categories: 1) the amount of delay between decoders, and 2) the type of information passed between decoders. We are primarily concerned with the first issue, although it is not entirely separate from the second.

Based upon the fact that we are considering trellis codes exclusively, the multistage decoder consists of a number of Viterbi decoders, one for each component code. The full-delay multistage decoder thus requires a delay between two stages equivalent to the delay of the Viterbi decoder for the earlier stage, which we set at six times the constraint length of the code. Our method of reducing this delay is to have the first decoder release bits early for use by the second decoder, while waiting the full delay to release bits to the output stream. The performance of this scheme varies with the amount of delay.

We investigate a simple approach using the latter of the two previous examples, in which the second decoder, instead of using subset metrics, uses the output of the first decoder to choose a signal point from each subset for use in calculating the distances. To establish the benchmark for this approach, we simulate the case in which the second decoder is given perfect knowledge of the input sequence to the first encoder. This is equivalent to making the assumption that the first decoder never makes an error in the bits it passes to the second decoder. The benchmark result, along with results for a range of delays in the first decoder, is shown in Figure 6.

Obviously, the longer the decoder waits before releasing the bits, the more reliable they will be. Yet, no matter what the chosen delay, we can further improve this approach by passing reliability information along with the bits. This reliability information might then used by the second decoder to soften the decisions passed to it by the first decoder. Following Hagenauer [5] we propose to use the path metric difference, on which the decisions of the first decoder are based, as the foundation of a simple measure of reliability. This is quite intuitive, for if the difference is zero, then the Viterbi Algorithm is forced to select one of the two paths arbitrarily, producing a totally unreliable decision.

In summary, the relationship between decoding delay and performance for multilevel codes is explored in this paper. Various trade-offs are investigated to show that for moderate performance penalties the amount of delay can be reduced significantly. Several decoding approaches are described, with simulation results presented to characterize the performance of each.

REFERENCES

[1] A. R. Calderbank and N. J. A. Sloane, "New Trellis Codes Based on Lattices and Cosets," *IEEE Transactions on Information Theory*, vol. 33, no. 2, pp. 177–195, March 1987.

[2] A. R. Calderbank, "Multilevel Codes and Multistage Decoding," *IEEE Transactions on Communications*, vol. 37, no. 5, pp. 222–229, March 1989.

[3] G. D. Forney, Jr., "Coset Codes– Part I: Introduction and Geometrical Classification," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1123–1151, September 1988.

[4] G. D. Forney, Jr., "Coset Codes– Part II: Binary Lattices and Related Codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1152–1187, September 1988.

[5] J. Hagenauer and P. Hoeher, "A Viterbi Algorithm with Soft Decision Outputs and Its Applications," *GLOBECOM '89 Conference Record*, pp. 47.1.1–47.1.7, November 1989.

[6] H. Imai and S. Hirakawa, "A New Multilevel Coding Method Using Error Correcting Codes," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 371–377, May 1977.

[7] G. J. Pottie and D. P. Taylor, "Multilevel Codes Based on Partitioning," *IEEE Transactions on Information Theory*, vol. 35, no. 1, pp. 87–98, January 1989.

[8] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 55–67, January 1982.

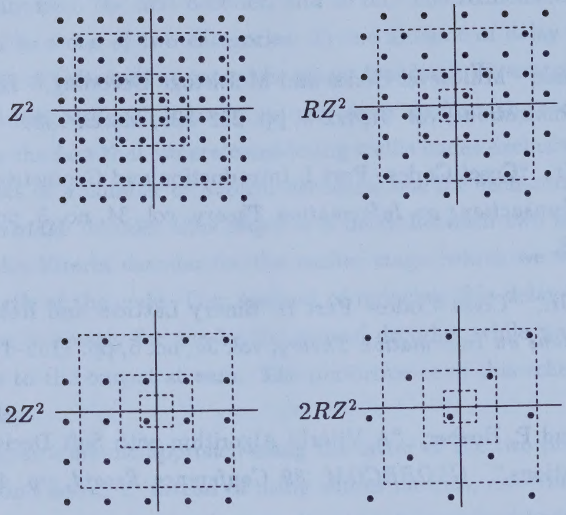

Z^2
RZ^2
$2Z^2$
$2RZ^2$

Figure 1: The Lattice Chain $Z^2/RZ^2/2Z^2/2RZ^2$

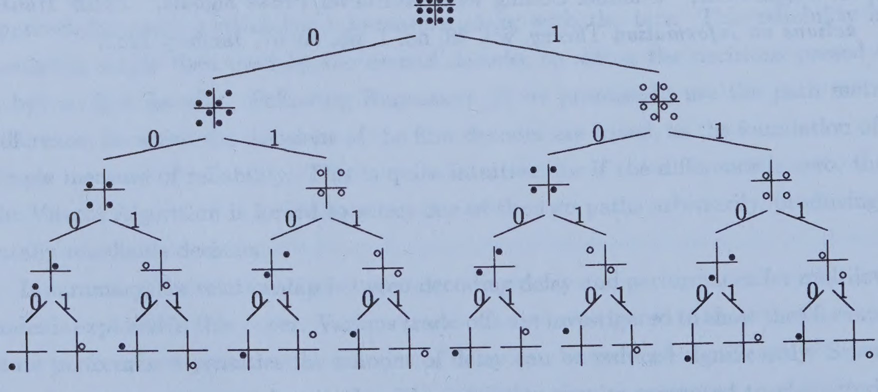


Figure 2: Subset Partitioning of 16-QAM

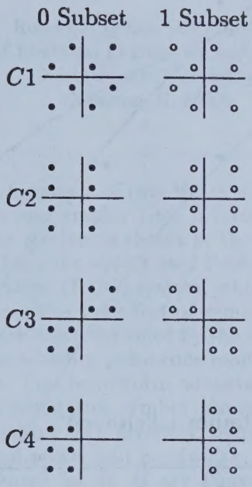

0 Subset
1 Subset
C1
C2
C3
C4

Figure 3: Subsets for Independent Decoding

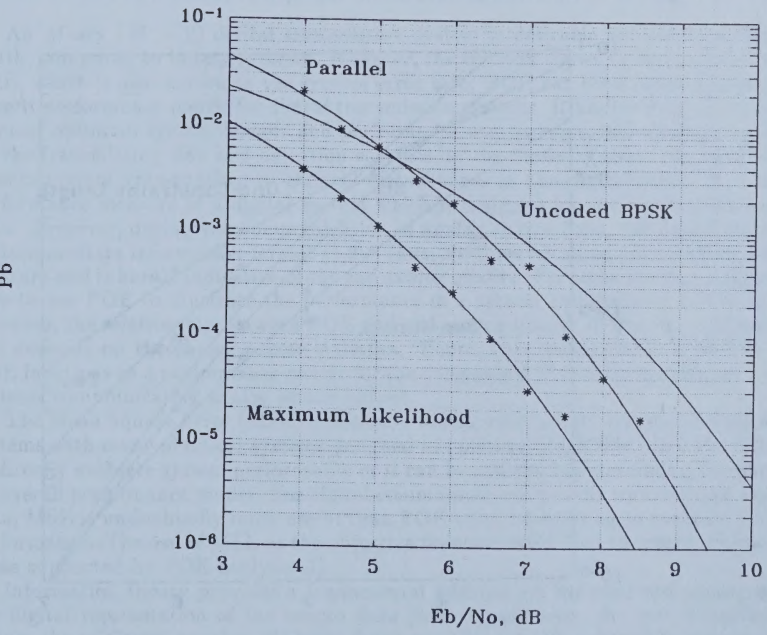


Figure 4: Performance of Two 2-State Codes

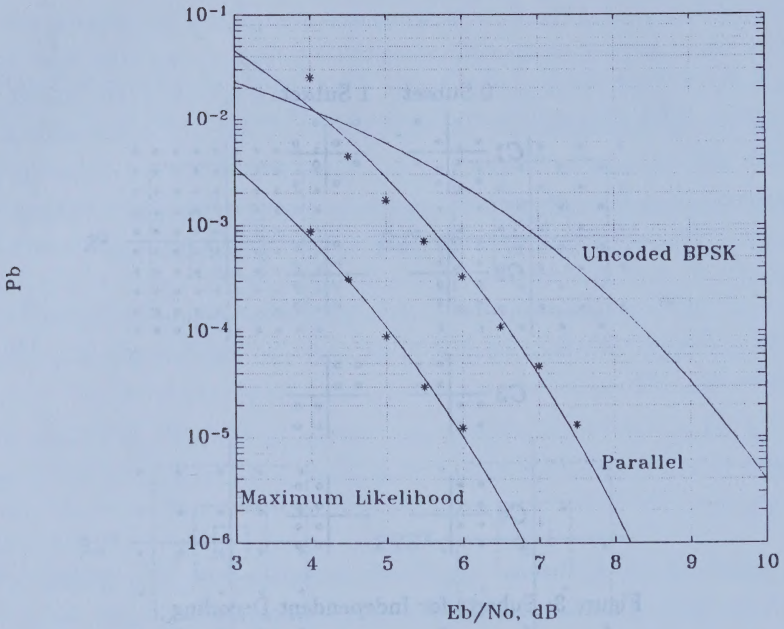


Figure 5: Performance of 4-State Code/2-State Code

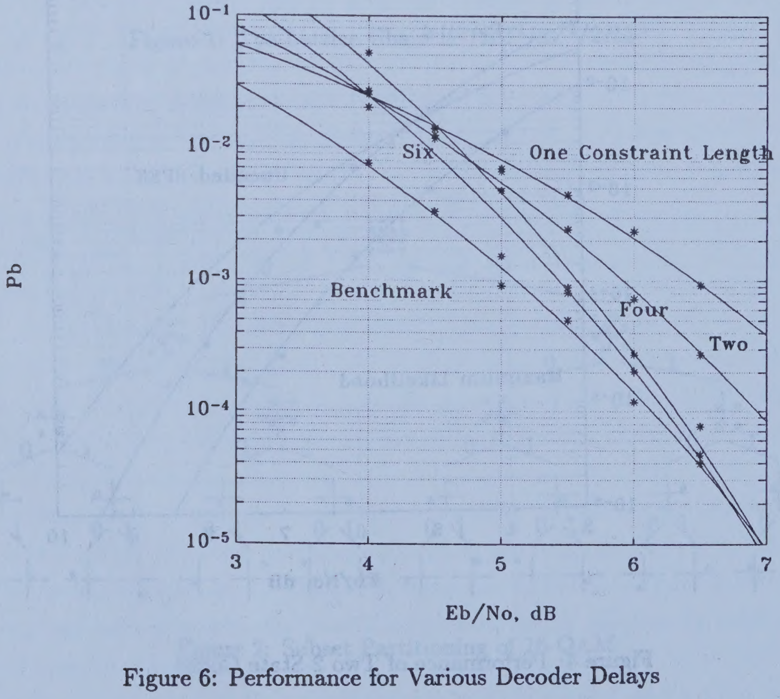


Figure 6: Performance for Various Decoder Delays

Digital Transmission of Amplitude Coded Signals with Diversity

Robert J. O'Dea and C.C. Lee
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208

Abstract

This paper examines the performance of two M-ary digital communication systems which transmit analog source samples and employ time diversity to combat pulse jamming. The Mean Square Error (MSE) at the receiver is chosen as the performance metric to evaluate end analog reception quality, rather than the widely used Probability of Error (POE) criterion. An M-ary Pulse Amplitude Modulation (PAM) system, which modulates the carrier amplitude with analog source samples, minimizes the effect of symbol errors on the overall MSE at the receiver. This performance feature is accompanied by the associated bandwidth efficiency of an M-ary system when compared to a binary pulse code modulation (PCM) system transmitting the same analog source samples. This bandwidth advantage can be exploited to provide time diversity to the system by subdividing each symbol into equal energy subsymbols, or chips. It is shown that an M-ary PAM system with diversity, significantly outperforms a binary PCM system when both consume equal power and bandwidth, and are subject to pulsed jamming noise. Similar results are presented for an M-ary Phase Shift Keyed (PSK) system which modulates the carrier phase with the analog source samples. The effect of the analog source distribution on the MSE performance is also discussed.

1. Introduction

An M-ary ($M > 2$) digital transmission system is desirable because it conserves bandwidth, compared to binary systems. However, the tradeoff shows up in the form of degraded POE, which is also known as the symbol error rate. POE has been commonly accepted as a generic performance metric for digital transmission systems. It has been the objective function in most optimum system designs and performance comparisons, including coding algorithms on the transmitting side and detection schemes on the receiving side. Essentially, POE represents system vulnerability to channel disturbances at the digital level. It is an effective performance measure of a digital system which is designed for the communication of digital data. However, during the communication of analog source data, the digital data is merely an intermediate information format [1,2]. Thus, POE serves as an intermediate performance measure and is hardly indicative of the end analog quality. For some systems it may be acceptable to use POE to illustrate the performance of a system transmitting analog information. However, the relationship between POE and end analog quality differs from system to system and depends on the analog source statistics. Hence, it is questionable as to how effectively POE functions as a performance metric for the comparison of two distinct digital transmission systems communicating analog source output.

The Mean Square Error (MSE) criterion is widely used in both communication and control systems with many practical systems designed to optimize MSE. The MSE can either be used to directly evaluate system performance or it can be utilized in determining key parameters in an overall performance model. For digital communication systems transmitting analog source data, MSE is undoubtedly more useful than POE when attempting to evaluate global system performance. The use of MSE as the objective function could lead to results which differ from those supported by POE analysis [7].

Information theory provides a fundamental relationship between end analog quality and the digital represetation of the source data [3-6]. Specifically, the rate distortion function defines the minimum number of binary digits per source letter required to encode that source so that it can be reconstructed to satisfy a given fidelity criterion. However, even if a source encoder achieves the the rate distortion bound, the binary data must be transmitted to the receiver without error to quarantee the fidelity criterion. For a practical system this requires the use of an effective channel code to allow transmission of this data at an error rate approaching zero. Such codes can be difficult to design and implement. Nevertheless the rate distortion function can be used in combination with the channel capacity to establish the theoretical optimum performance level possible for a given transmission system and source [7].

M-ary PAM system can be configured to use quantized source samples as the M transmitted amplitude levels. Such a system maintains a bandwidth advantage proportional to $log_2 M$ over a binary PCM system which encodes the same source samples before transmission. However the M-ary system suffers from an increase in POE as M increases. Consequently, if POE is the objective function, the PCM system would be more desirable. On the other hand, if MSE is the objective function and the noise statistics are Gaussian, the M-ary PAM system can outperform the PCM system; this is on top of the bandwidth advantage associated with the M-ary PAM system [7]. This behavior is direct result of the fact that the MSE of the M-ary PAM system is not effected as adversely by symbol errors. Specifically, if the noise statistics are Gaussian, it is straightforward to show that the mean-squared amplitude error caused by a symbol error is dominated by its smallest possible value. On the other hand, symbol errors in the received PCM signal can cause large mean-squared amplitude error if they occur in higher order code bits. It will also be shown that an M-ary PSK system exhibits similar MSE performance features as the M-ary PAM system, but to a lesser extent.

The MSE plotted as a function of bit energy-to-noise ratio illustrates the performance provided by various digital transmission systems for communication of analog signals. However, it fails to display the bandwidth advantage of M-ary systems. A more comprehensive comparison would result if the systems used the same energy and consumed the same bandwidth. Applying a repeat code of size $log_2 M$ to the M-ary PAM and PSK systems, while maintaining the same source data rate, would increase the signalling rate and cause the two M-ary systems to consume the same bandwidth as the binary PCM system. It must then be shown that the bandwidth advantage of the M-ary PAM and PSK systems can be converted to an MSE performance advantage when compared to the binary PCM system. The repeat code should yield a performance gain if it were employed with an interleaver and used to combat pulsed jamming noise [8,9]. The interleaver will guarantee that each of the received code bits, also known as "chips", has an independent chance of being affected by the pulsed jammer. In this configuration, the M-ary PAM and PSK systems, with repeat code and interleaver, are able to achieve time diversity.

The introduction of time diversity into a communication system has proven to be beneficial in countering both pulsed noise and multipath effects. Thus, evaluating the MSE performance of repeat coded M-ary PAM and PSK systems and comparing it to the performance of the PCM system when a pulse jammer is present, would lead to a bandwidth-related advantage for the M-ary systems. Specifically, the M-ary systems are able to employ a repeat code to combat a pulsed jammer while consuming the same bandwidth as a binary PCM system. The paper concludes with a discussion of the effect that the source distribution has on the MSE performance of these systems.

2. Performance Analysis

The amplitude samples to be transmitted are assumed to come from a uniformly distributed analog source X and are quantized via a q-level uniform quantizer with inter-level spacing of $1/q$. The source distribution is defined for $|x(t)| \leq 1$ by:

$$p\{x(t) = a\} = \frac{1}{2}, \qquad -1 \leq a \leq 1.$$

Samples of the source are assumed to be independent and thus the quantizer output levels become equally likely with probability $1/q$. The quantization error, which is equal to $1/3q^2$ for a uniform source, will be ignored since it is the same for all systems examined and can simply be added to the MSE caused by the transmission of the quantized amplitude samples.

M-ary PAM:

The signalling waveforms for an M-ary PAM system are represented in the form:

$$s_m(t) = A_m Re[u(t)e^{j2\pi f_c t}] \qquad m = 1, 2, \ldots M, \qquad 0 \leq t \leq T \tag{1}$$

where, without loss of generality, the signal amplitude A_m are assumed to take on the discrete values

$$A_m = (2m - 1 - M) \qquad m = 1, 2, \ldots, M$$

and $u(t)$ represents the equivalent low-pass pulse waveform, f_c the carrier frequency. By setting the number of signal levels equal to the number of quantizer levels ($M{=}q$), the M-ary PAM system can perform a one-to-one mapping of quantizer levels q_m to signal amplitudes A_m. Each transmitted symbol has energy,

$$E_m = \frac{1}{2} A_m^2 \int_0^T |u(t)|^2 \, dt \;\equiv\; A_m^2 \, E_u \; .$$

If $s_m(t)$ is transmitted over an AWGN channel, then the received signal $r(t)$, expressed in equivalent low pass form is

$$r(t) = \alpha e^{-j\phi} A_m u(t) + z(t).$$

where α represents the channel attenuation factor, ϕ accounts for any phase delay introduced during transmission and $z(t)$ represents the equivalent low-pass additive Gaussian noise. Assuming the receiver employs a matched filter and without loss of generality, $\phi = 0°$ and $\alpha = 1$, the decision variable becomes:

$$U = 2 E_u \, A_m + v \;\equiv\; u_m + v \; .$$

The Gaussian noise component v has zero mean and variance $\sigma_v^2 = 2 E_u \, N_0$, where N_o denotes the single sided noise spectral density. Therefore U is Gaussian with mean u_m and variance σ_v^2. The calculation of the MSE is based on the difference in magnitude between the transmitted level and the decoded level. Representing the quantizer levels at the source and sink by q_i and q_j respectively, and the probability of decoding level q_j when q_i was sent by $P_{j/i}$ yields:

$$\mathrm{MSE} = \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{M} [q_j - q_i]^2 \, P_{j/i}. \tag{2}$$

The conditional probability $P_{j/i}$ is a function of the pulse energy to noise ratio, E_u/N_o. To facilitate performance comparisons between transmission systems the MSE should be expressed in terms of the average bit energy-to-noise ratio, γ_b. Using the fact that $log_2 M$ bits can be transmitted via one M-ary symbol the average bit energy-to-noise ratio is given by

$$\gamma_b \;=\; \frac{ST}{N_o \, log_2 \, M} \tag{3}$$

where S denotes the average signal power:

$$S \;=\; \frac{E(A_m^2)}{T} \, E_\mu \tag{4}$$

with

$$E(A_m^2) \;=\; E[(2m - 1 - M)^2] \;=\; \frac{M^2 - 1}{3}. \tag{5}$$

Substituting (4) and (5) into (3) yields:

$$\gamma_b = \frac{M^2 - 1}{N_o log_2 M} \frac{E_u}{N_o}.$$

M-ary PSK:

The signalling waveforms for an M-ary PSK system transmitting the quantized source samples are represented in the form

$$s_m(t) \;=\; A \, Re[u(t) e^{j(2\pi f_c t + \phi_m)}] \qquad 0 \le t \le T$$

where

$$\phi_m = \frac{(2m-1-M)}{\pi} \qquad m = 1, 2, \ldots, M.$$

Once again each of the M quantizer levels q_m is mapped one-to-one to a phase angle ϕ_m. Each transmitted symbol has energy,

$$E_m = \frac{1}{2} A^2 \int_0^T |u(t)|^2 \, dt \equiv A^2 \, E_u.$$

If an optimum correlation receiver is used and AWGN with single sided spectral density N_o is assumed, the receiver estimate of the phase $\hat{\phi}$, is a random variable having a symmetrical distribution centered at the transmitted phase ϕ_m. This distribution is equivalent to the probability distribution of the phase for a sinusoid plus bandpass noise. Assuming for the moment that $\phi_m = 0°$, the distribution of the decoded phase $\hat{\phi}$ will be centered about 0° and given by:

$$p_{\hat{\phi}}(\hat{\phi}) = \frac{1}{2\pi} e^{-\frac{A^2}{2\sigma^2}} + \frac{A \cos \hat{\phi}}{\sqrt{2\pi\sigma^2}} \, exp(-\frac{A^2 sin^2 \hat{\phi}}{2\sigma^2})[1 - q(\frac{A \cos \hat{\phi}}{\sigma})], \qquad (6)$$

where $\sigma^2 = N_o/T$ and $-\pi < \hat{\phi} < \pi$. The probability distribution for other transmitted ϕ_m is derived by simply centering the distribution given above about ϕ_m. The MSE for M-ary PSK is evaluated with equation (2) where the transmitted quantizer level is represented by q_i and the decoded level by q_j. The conditional probability of decoding level q_j when q_i was transmitted is represented by $P_{j/i}$ and can be determined from the probability distribution of the phase angle at the receiver, equation (6). The MSE can then be calculated as function of the bit energy-to-noise ratio:

$$\gamma_b = \frac{ST}{N_o \, log_2 \, M}.$$

Binary PCM:

A binary PCM system utilizes a binary code to represent the quantized source samples before transmitting them via a binary transmission system. A binary Amplitude Shift Keyed (ASK) signal is utilized to transmit the pulse code bits, and its signalling waverforms are represented by:

$$s_m(t) = A_m s(t) = A_m Re[u(t) e^{j2\pi f_c t}] \qquad m = 1, 2 \qquad 0 \le t \le T,$$

where $A_1 = -A_2 = 1$. The probability of bit error for such a system is given by [10],

$$P_b = Q(\sqrt{\frac{2E_\mu}{N_0}}) = Q(\sqrt{2\gamma_b}).$$

A "natural" binary codeword of length $L = log_2 \, M$ is used to represent the M quantizer levels and the MSE is defined as:

$$\text{MSE} = \frac{4}{M^3} \sum_{q_i} \sum_{\bar{c}} (q_c - q_i)^2 \, P_{c/i}$$

where $\bar{c} = c_{L-1}, c_{L-2}, \ldots, c_0$ represents the code bits, q_c represents the decimal equivalent of $\bar{c}$ increased by one (or the decoded quantizer level), and $P_{c/i}$ is the probability of receiving codeword $\bar{c}$ given that the codeword corresponding to level i was transmitted. These transition probabilities are given by:

$$P_{c/i} = \prod_{k=0}^{L-1} P_k$$

where $P_k = P_b$ if c_k is in error and $P_k = 1 - P_b$ otherwise.

The MSE performance as a function of the average bit energy-to-noise ratio γ_b, is plotted in Figure 1 for M-ary PAM, M-ary PSK and binary PCM, where $M=q=128$. The curves show that although the M-ary systems have larger POE than the binary PCM system, they are able to perform quite well when the systems are evaluated for MSE. The PAM system actually performs better than the PCM system for $\gamma_b <6$ dB. As pointed out previously, such a behavior is due to the distribution of the symbol errors for the 128-ary PAM and PSK systems. For a given γ_b, the probability of symbol error is greater for the M-ary PAM system than for the PCM system. However, due to the Gaussian distribution of the decoded signal amplitude at the receiver, the errors are most likely to be the signal amplitudes immediately adjacent to the transmitted level. These errors contribute only moderately towards the overall MSE. Meanwhile, the PCM system has a probability of symbol error that affects each of the code bits equally. If one of the more significant bits is effected, a large error occurs between the transmitted and received levels. Such an error would have a large effect on the overall MSE. As γ_b increases the POE for the PCM system decreases quite rapidly and leads to a very low probability of incorrectly receiving transmitted PCM codewords and hence a small MSE for the PCM system. Meanwhile, since the signal energy in the M-ary PAM system is divided amongst a larger number of signal levels, the POE does not decrease quite so rapidly as γ_b increases. Thus, the MSE for the M-ary PAM system will be greater then the MSE for the PCM system for larger values of γ_b.

The M-ary PSK system benefits from a similar error distribution for the decoded phase $\hat{\phi}$. This is verified by checking the distribution of ϕ_m at the receiver with equation (6). Although it is not Gaussian distributed like the amplitude of the M-ary PAM system, it does cause symbol errors to be concentrated around the transmitted phase angle. One reason for its worse performance when compared to M-ary PAM can be attributed to the circular symmetry of the signal phase angles. The angles ϕ_1 and ϕ_{128}, which represent the smallest and largest quantizer levels respectively, are adjacent to each other in the phase space. Thus, whenever these two levels are transmitted, larger contributions to the overall MSE occur since the receiver has a significant probability of decoding the transmitted phase ϕ_1 as ϕ_{128} and vice versa. A "smart" mapping of the quantized amplitude levels to the phase angles might alleviate this effect. For instance, the smallest and largest quantizer levels could be represented by phase angles that are separated by 180°. While such a "smart" mapping may reduce the MSE associatd with the transmission of the smallest and largest quantizer levels, it will also have an effect on the MSE associated with other transmitted levels.

3. M-ary PAM and PSK with Diversity:

Thus far, the performance analysis has been conducted for systems with equal average power, but unequal bandwidth. Specifically, when transmitting the same quantized source samples, the binary PCM system utilizes $\log_2 M$ times more bandwidth than the M-ary systems. Increasing the signalling rate of the M-ary systems by a factor of $\log_2 M$ will increase the bandwidth of the M-ary systems so that it is equal to the bandwidth of the binary PCM system. Evaluating the MSE performance under conditions that allow the increased signalling rate of the M-ary systems to provide some performance advantage over the PCM system would help illustrate the value of the bandwidth efficiency that the M-ary systems possess.

By employing a repeat code which generates $\log_2 M$ code bits for each M-ary transmitted symbol, the signalling rate and thus the bandwidth would increase by a factor of $\log_2 M$. If these code bits were then distributed temporally, the system would achieve time diversity. Such a system could operate with the same power, bandwidth and source rate as a binary PCM system while providing some immunity to pulsed interference. This interference could be caused by a pulsed broadband Gaussian jammer. Such a jammer can do tremendous damage to a transmission system by concentrating all its power into short duration pulses. However, time diversity can be used to combat this effect and limit performance loss. Utilizing an M-ary PAM or PSK system with repeat code and interleaver should offer some performance advantage over a binary PCM system consuming equal bandwidth and power in the presence of pulsed interference.

The systems to be examined are depicted in Figure 2. Once again the data to be transmitted comes from a uniform q-level quantizer operating on a uniformly distributed source. The quantizer generates data at a rate r. For the binary PCM system these quantized amplitude samples are transmitted as described previously. However, for the M-ary systems the quantized amplitude sample is divided into $\log_2 M$ subsymbols, or chips. The duration of each chip is equal to,

$$T_c = \frac{1}{r log_2 M} = \frac{T}{log_2 M}.$$

The $\log_2 M$ chips corresponding to one quantizer output can be interpreted as a repetition code. These chips are then processed by an interleaver so that each chip has an independent chance of being affected by the pulsed noise. The chips are then transmitted via M-ary PAM or M-ary PSK modulation just as was previously described in this paper, with the exception that the duration of each signal pulse will be T_c rather than T. Assuming that the same average power S remains constant, the average energy in each chip is a fraction $1/\log_2 M$ of the average energy for each quantizer output.

The channel is subject to both channel AWGN and the pulsed broadband Gaussian noise supplied by a pulse jammer. In order to focus on the effects of the pulsed interference, it will be assumed that the bit energy-to-channel-noise ratio is 60dB and the power of the channel AWGN N_o, is kept negligible when compared to jamming noise. The jammer noise power thus becomes the determining factor when evaluating the MSE at the receiver. The MSE will be plotted as a function of the bit energy-to-jammer-noise ratio, γ_b^j.

The pulse jammer operates with $Prob$(jammer "on") $= \rho$, and $Prob$(jammer "off") $= 1-\rho$, for $0 \leq \rho \leq 1$. Assuming that the average jammer noise power is N_j, the jammer will affect the transmitted signal with noise of power N_j/ρ when the jammer is "on" and zero noise power when the jammer is "off". Finally it is assumed that a transmitted chip encounters either a channel with a jammer "on" condition, or a jammer "off" condition. Other cases are not investigated here. The probability of symbol error can then be written as:

$$Prob\{\text{symbol error}\} \quad = \quad \rho \cdot Prob\{\text{error/jammer "on"}\}.$$

The above equality is a result of the assumption that the power of the channel AWGN is negligible. This same principle is applied to the calculation of the conditional probabilities $P_{j/i}$, which are needed to determine the MSE. Upon deinterleaving the received chips, a "majority" rule is employed to estimate the transmitted quantizer level.

The PCM system is effected by the pulse jammer in a similar fashion as the M-ary systems, since each binary code bit in the PCM system has the same duration T_c, as one of the chips transmitted in the M-ary systems. The probability of bit error can be written as:

$$P_b = \rho \cdot Q(\sqrt{\frac{2\rho E_u}{N_j}}).$$

Figures 3a-c, display the MSE performance of binary PCM, M-ary PAM and M-ary PSK, respectively. The MSE is plotted as a function of the bit energy-to-jammer noise ratio γ_b^j, for various ρ. Examining the plot for PCM (Figure 3a), shows that when the pulsed noise power is quite large (γ_b^j small), the MSE increases as ρ increases. This behavior arises because the pulse noise is powerful enough to cause errors in the transmitted PCM bits even when it is spread out in time ($\rho \longrightarrow 1.0$). Thus, as the likelihood of the noise being present increases, so does the number of transmitted bits being affected by noise, leading to an increased MSE. On the other hand, as the bit energy-to-jammer noise ratio increases, the receiver can more likely decode the transmitted bits correctly, and thus the MSE decreases. However, it can be seen that as the noise becomes concentrated in shorter bursts ($\rho \longrightarrow 0$), the receiver has trouble decoding bits during this burst. If these errors occur in high order bits, they will contribute significantly to the overall MSE. Therefore, for large γ_b^j, shorter duration pulsed noise has a more pronounced effect in degrading the MSE performance of the PCM system.

Figure 3b shows that the PAM system with diversity is not so adversely affected by the pulse interference. Due to the diversity provided by the repeat code and interleaver, the MSE actually decreases as ρ decreases. The probability that the pulsed noise affects a majority of the transmitted chips is decreased as ρ decreases. Also, because the symbols represent actual quantized amplitude samples, and the noise is Gaussian, most errors caused by the pulse noise are concentrated in the quantizer levels nearest the transmitted level. Thus, for large γ_b^j, the jammer is not able to increase the MSE by concentrating its noise power on a smaller number of transmitted symbols, as is the case with a binary PCM system.

Finally, Figure 3c shows that the repeat coded M-ary PSK system has a similar ability to combat pulsed noise as the M-ary PAM system. It too benefits from the diversity provided by the repeat code and interleaver. The probability that the pulse jammer can affect a majority of the chips representing a transmitted quantizer level decreases as ρ decreases. Due to the phase probability distribution at the receiver, the PSK system is also able to limit decoder errors to phase angles that are closer to the transmitted phase angle. This reduces the amount of MSE generated by symbol errors caused by the pulsed interference noise.

Figures 4a and 4b contrast the MSE performance of the M-ary PAM and PSK systems with diversity, and the binary PCM system, for ρ=.3 and ρ=.1, respectively. It is shown that as ρ decreases the M-ary PAM and PSK systems experience improved MSE performance. These two figures also illustrate the performance discrepancies between the two M-ary systems. Due to the two distinct signal modulation techniques employed, the probability distribution for the transmitted symbols is different for the two demodulators. For large jammer noise power (small γ_b^j), the PSK system outperforms the PAM system because it is more successful in concentrating symbol errors to levels close to the transmitted level. However, as the jammer noise power decreases the MSE performance improves much more rapidly for the PAM system than for the PSK system. Since for large γ_b^j, incorrectly decoded symbols are usually located immediately adjacent to the transmitted symbol, the performance discrepancy is attributed to the circular symmetry of the phase angles in the PSK system. The error that occurs during transmission of the smallest and largest quantizer levels (ϕ_1 and ϕ_{128}), dominates the MSE performance since they are adjacent symbols and involve the largest magnitude error between quantizer levels. M-ary PAM does not suffer from such an effect.

4. Performance Effect of Source Statistics

All the previous analysis was conducted assuming a uniformly distributed source. It is of interest to investigate MSE performance for different source distributions. The generalized Gaussian distribution [11] is chosen for this work since it encompasses distributions of various tail decaying rates, its density function is given by

$$f_n(x) = [c\eta(\sigma_n, c)/2\Gamma(1/c)]exp\{-[\eta(\sigma_n, c) \mid x \mid]^c\}$$

where

$$\eta(\sigma_n, c) \equiv \sigma^{-1}[\Gamma(3/c)/\Gamma(1/c)]^{1/2}$$

and c is a positive parameter controlling the rate of decay, $\Gamma(\cdot)$ is the gamma function, and σ_n^2 is the noise variance. Choosing various values for the parameters c and σ generates distributions ranging from an impulse centered at zero (σ small), to a near uniform distribution (σ large).

Varying the parameters c and σ and evaluating the MSE performance of the M-ary PAM, M-ary PSK and binary PCM systems demonstrates that only the M-ary PSK system exhibits any significant performance change. For the M-ary PSK system the generalized Gaussian distribution with parameters $c = 10$ and $\sigma = .45$ causes the most noticeable shift in performance from that determined with a uniform source distribution. The generalized Gaussian distribution with these parameters is very similar to the uniform distribution with the exception that the density decreases smoothly but rapidly. The MSE performance corresponding to these two source distributions is shown in Figure 5 for the M-ary PSK system. The performance is much better for the generalized Gaussian distribution ($c = 10, \sigma = .45$), because the phase angles ϕ_1 and ϕ_{128} are rarely used. Due to their being adjacent in the signal format and the fact that they represent the smallest and largest quantizer levels respectively, these transmitted angles have a large MSE associated with them. Thus, reducing the probability of the smallest and largest quantizer levels significantly decreases the MSE for the M-ary PSK system.

The MSE associated with each transmitted level in the M-ary PAM system is relatively constant for all the levels. Hence, the M-ary PAM system is quite robust with respect to source distribution. For the binary PCM system, robustness can be attributed to the "natural" binary code. The code spreads the POE equally across all transmitted levels.

5. Conclusion

Results were presented to illustrate how the bandwidth advantage inherent to M-ary transmission systems can be converted to a performance gain when they operate in the presence of pulsed noise. Specifically, an M-ary PAM system, which modulates the carrier amplitude, and an M-ary PSK system, which modulates the carrier phase, were employed with time diversity to transmit quantized amplitude samples from an analog source. The MSE performance of these two systems was compared to a binary PCM system consuming equal amounts of power and bandwidth. MSE performance curves showed that the M-ary systems provided significant performance advantage over the PCM system.

References

[1] T.J. Goblick Jr., "Theoretical limitations on the transmission of data from analog sources," IEEE Trans. Inform. Theory, vol IT-11, pp. 558-567, October 1965.

[2] T.J. Goblick Jr. and J.L. Holsinger, "Analog Source Digitization: A Comparison of Theory and Practice," IEEE Trans. Inform. Theory, vol IT-13, pp 323-326, April 1967.

[3] C.E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, vol 37, pp. 379-623, 1948.

[4] C.E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," IRE Nat'l Conv. Rec., pt. 4, pp 142-163, 1959.

[5] R.G. Gallager, "Information Theory and Reliable Communication", Wiley, New York, 1968.

[6] R.E. Blahut, "Principles of Information Theory", Addison-Wesley, 1987.

[7] R.J. O'Dea and C.C. Lee, "Performance Comparison of Digital Transmission Systems for Analog Sources," Proceedings of 1990 Conference on Inf. Science and Systems, Princeton, NJ, March 1990. (to appear)

[8] R.L. Pickoltz, D.L. Schilling and L.B. Milstein, "Theory of Spread Spectrum Communications–A tutorial," IEEE Trans. Commun., COM-30, pp. 855-884, May 1982.

[9] M.K. Simon, J.K. Omura, R.A. Scholtz and B.K. Levitt, "Spread Spectrum Communications", vol. I, Computer Science Press, Inc., Rockville, MD, 1985.

[10] J.G. Proakis, "Digital Communications," 2nd ed., McGraw-Hill, New York,1989.

[11] J.H. Miller and J.B. Thomas, "Detectors for Discrete-Time Signals in Non-Parametric Gaussian Noise," IEEE Trans. Inform. Theory, vol IT-18, pp. 241-250, March 1972.

Acknowledgement

This research was supported in part by American Iron and Steel Institute (AISI) through the Steel Resource Center (SRC) at Northwestern University.

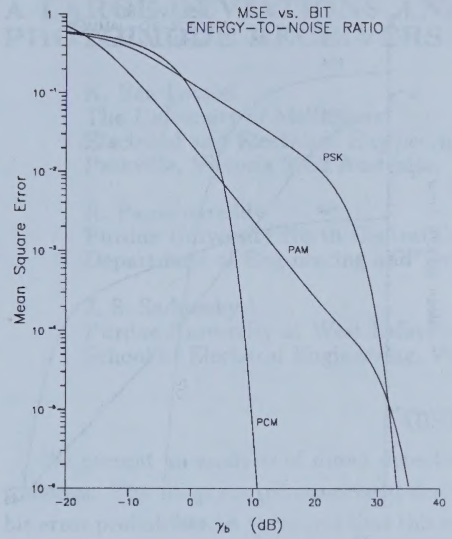


Figure 1. MSE for 128-ary PAM
128-ary PSK and
Binary PCM.

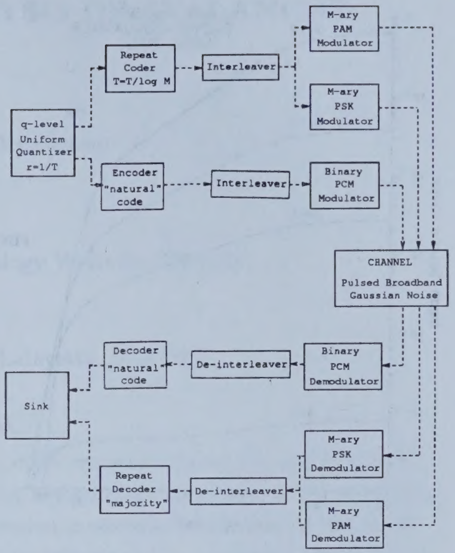

q-level
Uniform
Quantizer
r=1/T
Repeat
Coder
T-T/log M
Interleaver
M-ary
PAM
Modulator
M-ary
PSK
Modulator
Encoder
"natural"
code
Interleaver
Binary
PCM
Modulator
CHANNEL
Pulsed Broadband
Gaussian Noise
Decoder
"natural"
code
De-interleaver
Binary
PCM
Demodulator
Sink
M-ary
PSK
Demodulator
Repeat
Decoder
"majority"
De-interleaver
M-ary
PAM
Demodulator

Figure 2. 128-ary PAM with diversity,
128-ary PSK with diversity,
and Binary PCM systems.

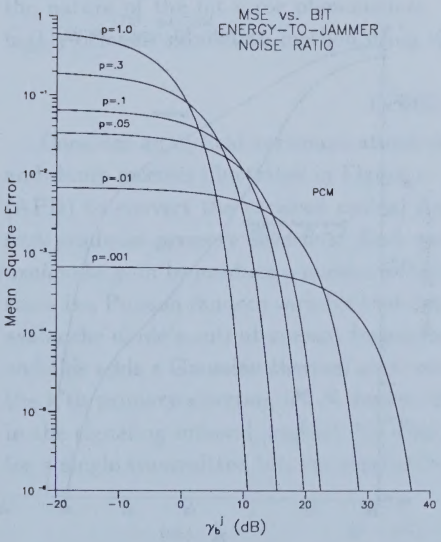


Figure 3a. MSE for Binary PCM
with Pulsed Noise.

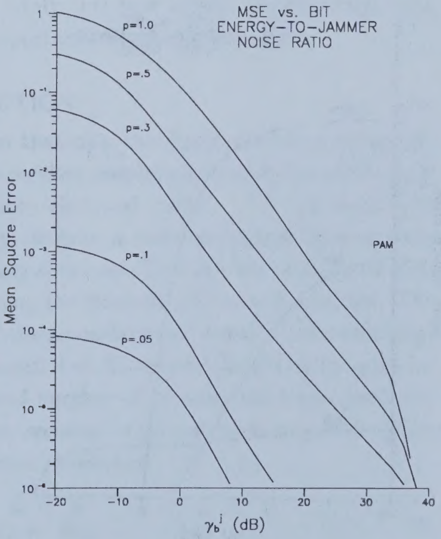


Figure 3b. MSE for 128-ary PAM
with Pulsed Noise.

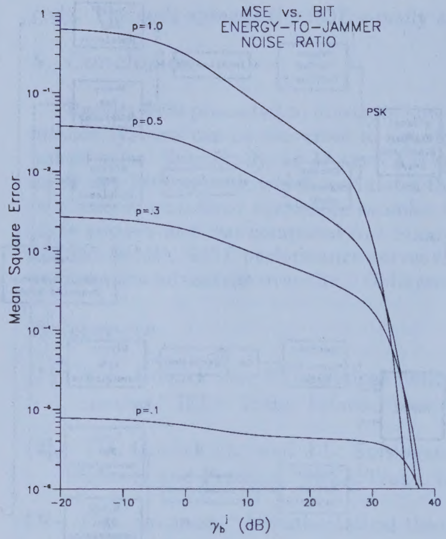


Figure 3c. MSE for 128-ary PSK with Pulsed Noise.

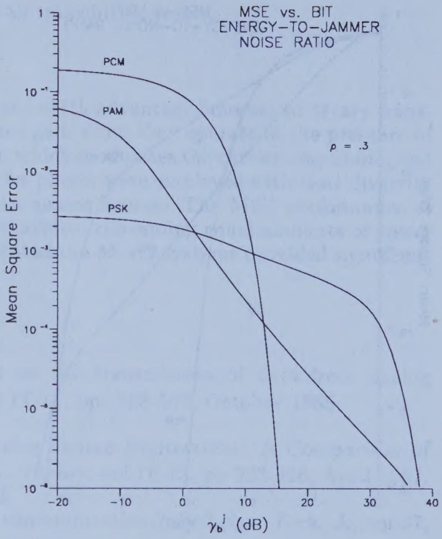


Figure 4a. 128-ary PAM with diversity, 128-ary PSK with diversity, Binary PCM in the presence of Pulsed Noise ($\rho = .3$).

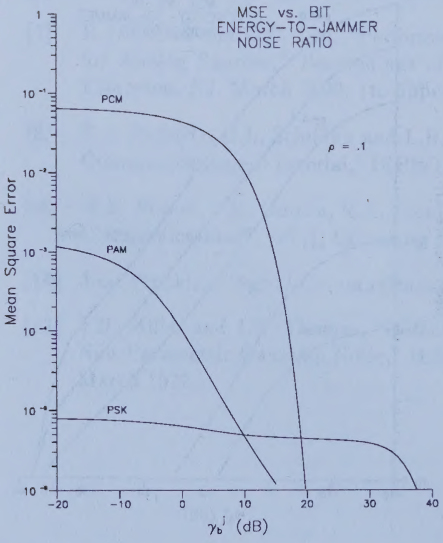


Figure 4b. 128-ary PAM with diversity, 128-ary PSK with diversity, Binary PCM in the presence of Pulsed Noise ($\rho = .1$).

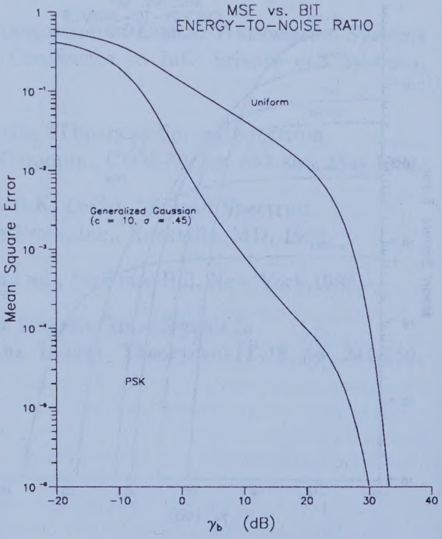


Figure 5. 128-ary PSK with Generalized Gaussian and Uniform source distributions.

A LARGE DEVIATIONS ANALYSIS OF AVALANCHE PHOTODIODE RECEIVERS

K. Ben Letaief
The University of Melbourne
Electrical and Electronic Engineering Department
Parkville, Victoria 3052 Australia

R. Papannareddy
Purdue University North Central Campus
Department of Engineering and Technology Westville, IN 46391

J. S. Sadowsky [1]
Purdue University at West Lafayette
School of Electrical Engineering, West Lafayette, IN 47907

ABSTRACT

We present an analysis of direct detection optical receivers employing avalanche photodiodes. The main contribution is to derive the asymptotic expression of the receiver's bit error probability. It turns out that this expression is essentially identical to the saddlepoint approximation, and hence, this result can be interpreted as a rigorous statement of the tightness of the saddlepoint methods. However, in contrast to the saddlepoint method (which is an analytical technique used to evaluate inverse Laplace transform integrals), the large deviations analysis presented here is "probabilistic" in nature. Our analysis is intimately linked to the probabilistic nature of the shot noise and avalanche gain models. As a result, the process of deriving the asymptotic expression sheds some new light on the nature of the bit error phenomenon. We apply this new insight to the design of a highly efficient simulation method using the importance sampling technique.

INTRODUCTION

Consider an optical communications system that uses the direct detection integrate-and-dump receiver illustrated in Figure 1. This receiver employs an avalanche photodiode (APD) to convert the received optical signal into electrical current. The incident light first produces *primary electrons*. Each primary electron is then multiplied by a random *avalanche gain* to produce a number of *secondary electrons*. The number of primary electrons is a Poisson random variable that depends on the intensity of the optical signal. The avalanche diode's output current (consisting of the secondary electrons) is then amplified and this adds a Gaussian thermal noise component. Let X_k denote the avalanche gain for the k'th primary electron, let N denote the total number of primary electrons produced in the signaling interval, and let N_G denote the receiver's thermal noise response. Then for a single transmitted bit, the receiver's decision statistic is

$$D = \sum_{k=1}^{N} X_k + N_G. \tag{1}$$

We assume OOK (on-off keying) modulation of the optical intensity. The statistical hypotheses are

---

[1] J. S. Sadowsky is supported by NSF (ECS-9003007).

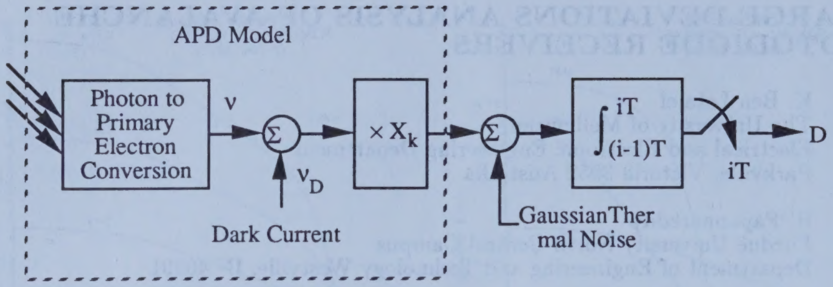


Figure 1: The integrate and dump APD receiver.

$$H_0 \text{ (0 transmitted): } N \text{ is Poisson with parameter } \lambda_0$$
$$H_1 \text{ (1 transmitted): } N \text{ is Poisson with parameter } \lambda_1$$

where $\lambda_0 < \lambda_1$. Under both hypotheses the avalanche gain sequence $\{X_k\}$ is i.i.d. (independent and identically distributed) [3, 4, 5]. Our goal is to estimate the bit error probabilities $P_0 = \mathcal{P}_0(\mathit{bit\ error}) = \mathcal{P}_0(D \geq \gamma)$ and $P_1 = \mathcal{P}_1(\mathit{bit\ error}) = \mathcal{P}_1(D \leq \gamma)$ where γ is the test threshold and $\mathcal{P}_i(\cdot)$ is the H_i probability distribution.

In this paper we shall attack the problem of estimating P_0 and P_1 using the asymptotic techniques of large deviations theory. Our asymptotic limit considers large Poisson intensities λ_0 and λ_1. Since we consider a limit as both λ_0 and $\lambda_1 \to \infty$, we must also control the threshold γ and the Gaussian noise variance in the limit. Actually, with no loss of generality, we shall take γ to be the primary control variable by setting $\lambda_i = c_i\gamma$ where c_i is a constant. From a practical point of view, this asymptotic analysis should be accurate when there is as few as 20 per bit. Our primary contribution is to prove the limit

$$P_i \sim \frac{C_i^*}{\sqrt{\gamma}} \exp(-I_i^*\gamma)$$

where C_i^* and I_i^* are positive constants. The symbol $\sim$ does not simply mean "approximation." We write $a_\gamma \sim b_\gamma$ to mean precisely that $\lim_{\gamma\to\infty} a_\gamma/b_\gamma = 1$. Thus, the above expression is a tight asymptotic result.

It turns out that the right side of our asymptotic expression is essentially identical to the saddlepoint approximation. Helstrom has provided some numerical computations indicating that the saddlepoint approximation can be very accurate for many practical systems [2]. One could interpret our asymptotic result as rigorous theoretical verification of the asymptotic tightness of the saddlepoint approximation. However, there is more to the large deviations approach than simply providing another vehicle for deriving a previously known expression. The saddlepoint technique is purely "analytical" in nature. It evaluates a line integral on the complex plain, and as far as the technique is concerned it is entirely irrelevant that the integral arises as an inverse Laplace transform of a moment generating function. In contrast, the large deviations approach is "probabilistic." It is intimately related to the nature of the shot noise and random avalanche gain model, and as a result, in the process of proving the asymptotic formula we gain some new and valuable insight into the behavior of the bit error phenomenon.

In addition to the asymptotic analysis, we can also compute the bit error probabilities to arbitrary precision using the Monte Carlo technique known as *importance sampling.* Space does not allow a full exposition of our importance sampling technique. We shall simply say that we have applied the large deviations theory of importance sampling developed in [1] and elsewhere, and present some numerical results.

The paper is organized as follows. Section 2 presents our large deviations analysis as a sequence of propositions leading to our main theorem. Section 3 presents some numerical computation using real avalanche gain distributions. Section 4 is the concluding section which discusses generalization of these results to consider ISI (intersymbol interference) and ISI equalizers.

LARGE DEVIATIONS ASYMPTOTIC ANALYSIS

We consider estimation of the bit error probabilities in the context of large λ_0 and λ_1 asymptotics. There are actually four parameters that must be controlled in the limit: λ_0, λ_1, $\sigma_G^2 = \mathrm{var}[N_G]$ and γ. To do this we must choose a single parameter to control the others. We choose γ. We shall consider the limit as $\gamma \to \infty$ with $\lambda_i = c_i\gamma$, and $\sigma_G^2 = c_G\gamma$ where c_0, c_1 and c_G are positive constants. For consistent detection (that is, in order to have vanishing bit error probabilities as $\gamma \to \infty$) we require that $E_0[D] < \gamma < E_1[D]$. Since $E_i[D] = \lambda_i\mu_X$ where $\mu_X = E[X_k]$, this implies that we must have $c_0 < \mu_X^{-1} < c_1$.

We wish to investigate the asymptotic behavior of $P_{i,\gamma} = \mathcal{P}_{i,\gamma}(bit\,error)$. Note that we have now augmented the notation with the subscript γ to emphasize the role of γ as the control variable. We begin by writing

$$P_{i,\gamma} \;=\; \sum_{n=0}^{\infty} q_{i,\gamma}(n)\, p_{i,\gamma}(n) \tag{2}$$

where $p_{i,\gamma}(n) = (c_i\gamma)^n e^{-c_i\gamma}/n!$ is the Poisson probability of the event $\{N = n\}$, and

$$q_{0,\gamma}(n) \;=\; \mathcal{P}\left(\sum_{k=1}^{n} X_k \,+\, N_G \geq \gamma\right)$$

and $q_{1,\gamma}(n)$ is as above with the "$\geq$" sign replaced by "$\leq$." $q_{0,\gamma}(n)$ and $q_{0,\gamma}(n)$ are the conditional bit error probabilities given $\{N = n\}$.

Consider the sum (2). Momentarily, for discussion consider only H_0 and suppose $\sigma_G^2 = 0$. We identify two effects that may combine to cause bit errors: 1) unusually large N, and 2) unusually large X_k's. In the case of the pure Poisson problem ($X_k \equiv$ constant $= \mu_X$) only the first effect occurs. In this case, $q_{0,\gamma}(n) = 0$ for $n < \gamma/\mu_X$ and $q_{0,\gamma}(n) = 1$ for $n > \gamma/\mu_X$, and it follows that the sum (2) is dominated by values of n near γ/μ_X. Now, suppose $X_k \not\equiv$ constant. Instead of having $q_{0,\gamma}(n)$ change abruptly from 0 to 1 at $n = \gamma/\mu_X$, there is a gradual transition located in the vicinity of $n = \gamma/\mu_X$. The end result is that the critical values of n, that is, the values of n that dominate in (2), may be somewhat less than γ/μ_X. But in any case, *these dominate values of n are roughly proportional to γ, and hence, these dominate values drift off to ∞ in the limit as $\gamma \to \infty$.* Likewise, a similar effect occurs for H_1 with the dominate values of n being somewhat larger than γ/μ_X.

Figure 2 illustrates the principle discussed above for H_0. In this example, X_k has exponential density $f(x) = e^{-x}$ for $x \geq 0$ (so $\mu_X = 1$), $c_0 = 1/2$ and $\sigma_G^2 = 0$. The

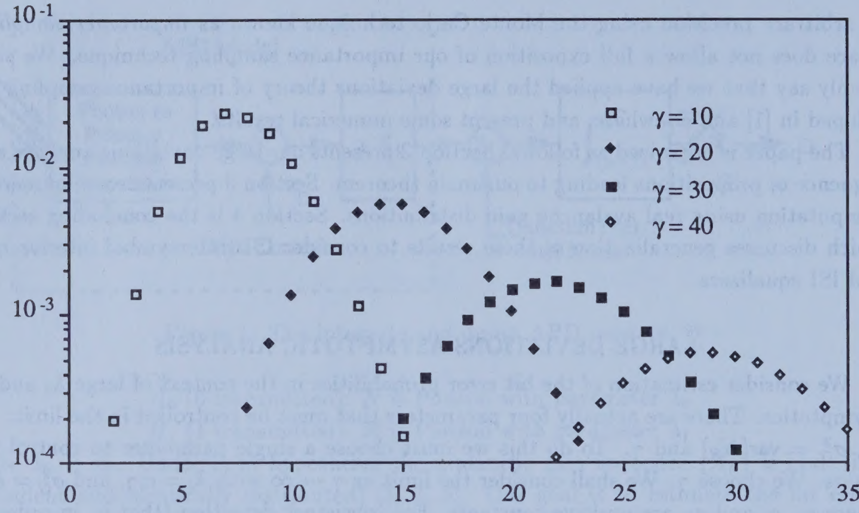


Figure 2: Behavior of $q_{0,\gamma}(n)p_{0,\gamma}(n)$ as a function of n.

product $q_{0,\gamma}(n)p_{0,\gamma}(n)$ is plotted as a function of n for $\gamma = 10$, 20, 30 and 40. Noting that the vertical axis is plotted on a logarithmic scale, we see that the general scale of these plots apparently decays exponentially with γ. We also see the "drifting off to ∞" of the dominate values of n discussed above. The maxima for these plots are $n = 7$, 14, 22 and 29 respectively for $\gamma = 10$, 20, 30 and 40. These maxima occur roughly at 0.7γ which is less than $\gamma/\mu_X = \gamma$.

In order to proceed with the analysis, we must first stabilize the "drifting off to ∞" effect. This stabilization will be achieved by a change of variables, but to do this we must first frame the problem in the proper setting. We rewrite the sum (2) as an integral:

$$P_{i,\gamma} = \sum_{n=0}^{\infty} q_{i,\gamma}(n)\, p_{i,\gamma}(n) = \int_0^{\infty} q_{i,\gamma}(\lfloor z \rfloor)\, p_{i,\gamma}(\lfloor z \rfloor) dz$$

where $\lfloor z \rfloor$ is the greatest integer $\leq z$. Note that the integrand is constant on the intervals $[n, n+1)$, $n = 0, 1, 2, \ldots$, and hence, there is no approximation introduced by this integral representation. Now we apply the change of variables $w = \gamma/z$. Define

$$\tilde{p}_{i,\gamma}(w) = p_{i,\gamma}(\lfloor \gamma/w \rfloor) = \frac{(c_i\gamma)^{\lfloor \gamma/w \rfloor}}{\lfloor \gamma/w \rfloor !}\, e^{-c_i\gamma}, \tag{3}$$

$$\tilde{q}_{0,\gamma}(w) = q_{0,\gamma}(\lfloor \gamma/w \rfloor) = \mathcal{P}\left(\sum_{k=1}^{\lfloor \gamma/w \rfloor} X_k + N_G \geq \gamma \right) \tag{4}$$

and $\tilde{q}_{1,\gamma}(w)$ is as above with the "$\geq$" sign replaced by "$\leq$." Then from the change of variables we obtain

$$P_{i,\gamma} = \int_0^{\infty} \tilde{q}_{i,\gamma}(w)\, \tilde{p}_{i,\gamma}(w)\, \frac{\gamma}{w^2}\, dw. \tag{5}$$

Equation (5) is the foundation of our analysis. We proceed as follows. First, for each fixed value of w we demonstrate that both $\tilde{p}_{i,\gamma}(w)$ and $\tilde{q}_{i,\gamma}(w)$ vanish exponentially fast as $\gamma \to \infty$ and that the exponential rates of decrease are stable functions of w. For large γ, (5) is dominated by values of w for which the exponential rate of decrease of the integrand is slowest. But as just noted, these rates are stable in w, and hence, the dominate values of w are asymptotically stable around some critical value w_i^* where the rate is minimized. Relating this stabilizing change of variables back to (2), the dominate values of n in the bit error probability sum are $n \approx \gamma / w_i^*$. This identification of the critical value w_i^* is a classical method of asymptotic analysis due to Laplace. Simply stated, "the exponential rate of decrease of an integral is equal to the minimum exponential rate of decrease of it's integrand." Thus, the critical value w_i^* not only determines the dominate region of integration in (5), but it also gives us the exponential behavior $P_{i,\gamma}$.

Define the log-moment generating function

$$\Lambda_X(\alpha) \;=\; \log\left(\mathrm{E}\left[e^{\alpha X_k}\right]\right).$$

Since $X_k \geq 0$, it turns out that the region of convergence $\{\alpha : \Lambda_X(\alpha) < \infty\}$ is an interval of the form $(-\infty, \bar{\alpha})$ or $(-\infty, \bar{\alpha}]$ for some $0 \leq \bar{\alpha} \leq \infty$. Moreover (assuming $X_k \not\equiv$ constant), $\Lambda_X(\alpha)$ is analytic, strictly increasing and strictly convex on $(-\infty, \bar{\alpha})$, $\Lambda_X(0) = 0$ and $\Lambda'_X(0) = \mu_X = \mathrm{E}[X_k]$. Next, we define $\alpha^*(w)$ implicitly as the solution of the equation

$$\Lambda'_X(\alpha^*(w)) \;=\; (1 - c_G \alpha^*(w))\, w, \tag{6}$$

and we define the *large deviations rate function* as

$$I(w) \;=\; \alpha^*(w)\, w \;-\; \left(\Lambda(\alpha^*(w)) + \frac{c_G\, w\, \alpha^*(w)^2}{2}\right). \tag{7}$$

Cramér's theorem is the fundamental large deviations theorem that identifies the exponential rate of decrease of the probabilities $\tilde{q}_{i,\gamma}(w)$ in terms of the function $I(w)$. This is why $I(w)$ is called a "rate function." Proposition 1 below characterizes definitions (6) and (7). Proposition 2 is the actual application of Cramér's theorem.

Proposition 1 *Assume $X_k \not\equiv$ constant, $\bar{\alpha} > 0$ and $\Lambda'_X(\alpha) \uparrow \infty$ as $\alpha \uparrow \bar{\alpha}$. Then for all $w \in (0, \infty)$ there exists a unique solution of (6); $\alpha^*(w)$ is analytic and strictly increasing on $(0, \infty)$ with $\alpha^*(\mu_X) = 0$; and $I(w)$ in (7) is a strictly convex, non-negative analytic function on $(0, \infty)$ with $I(w) = 0$ if and only if $w = \mu_X$.*

Proposition 2 *Assume the hypothesis of Proposition 1. If either $w > \mu_X$ for H_0 or $w < \mu_X$ for H_1, then as $\gamma \to \infty$ we have*

$$\tilde{q}_{i,\gamma}(w) \;\sim\; C(w)\sqrt{\frac{w}{\gamma}}\, \exp\left(-\Lambda_X(\alpha^*(w)) \bmod \left(\frac{\gamma}{w}\right)\right)\, \exp\left(-\frac{I(w)}{w}\gamma\right) \tag{8}$$

where

$$C(w) \;=\; \sqrt{\frac{1}{2\pi\,(\alpha^*(w))^2\,(\Lambda''_X(\alpha^*(w)) + c_G w)}}. \tag{9}$$

Otherwise, if either $w < \mu_X$ for H_0 or $w > \mu_X$ for H_1, then $\tilde{q}_{i,\gamma}(w) \to 1$.

In the first exponential factor, the "mod" function is $\mathrm{mod}(x) = x - \lfloor x \rfloor$, and hence, this factor simply oscillates between 1 and $\exp(-\Lambda_X(\alpha^*(w)))$. The most important factor in the asymptotic expression (8) is the second exponential factor. It vanishes with exponential rate $I(w)/w$. To underscore the importance of this factor, we write

$$\tilde{q}_{i,\gamma}(w) \quad \sim_{LD} \ \exp\left(- \frac{I(w)}{w} \gamma \right) \tag{10}$$

for $w > \mu_X$ under H_0 or $w < \mu_X$ under H_1. By $a_\gamma \sim_{LD} \exp(-r\gamma)$ we mean precisely that $\lim_{\gamma \to \infty} \log(a_\gamma)/\gamma = -r$. In words, "$\sim_{LD}$" means "asymptotically in the large deviations sense."

Next we consider the asymptotic behavior of the Poisson factor in the integrand of (5). This result is obtained from an application of Sterling's formula.

Proposition 3 *For all $w \in (0, \infty)$, as $\gamma \to \infty$ we have*

$$\tilde{p}_{i,\gamma}(w) \ \sim \ \sqrt{\frac{w}{2\pi\gamma}} \exp\left(-\log(c_i w) \ \mathrm{mod}\left(\frac{\gamma}{w}\right)\right)$$
$$\times \exp\left(- \left[\frac{(c_i w - 1) \ - \ \log(c_i w)}{w} \right] \gamma \right). \tag{11}$$

The dominate exponential part of (11) is

$$\tilde{p}_{i,\gamma}(w)) \quad \sim_{LD} \ \exp\left(- \left[\frac{(c_i w - 1) \ - \ \log(c_i w)}{w} \right] \gamma \right). \tag{12}$$

Notice that from $\log(x) \leq x - 1$ with $\log(x) = x - 1$ if and only if $x = 1$, it follows that the exponential rate of decrease of $\tilde{p}_{i,\gamma}(w)$ is non-negative and equals 0 if and only if $w = c_i^{-1}$.

In (10) and (12), we have identified the exponential rates of decrease of the factors $\tilde{q}_{i,\gamma}(w)$ and $\tilde{p}_{i,\gamma}(w)$. For each fixed w, the exponential rate of decrease of the product is $\tilde{q}_{i,\gamma}(w)\tilde{p}_{i,\gamma}(w)$ is the sum of the two corresponding rates. Thus, the integrand in (5) is

$$\tilde{q}_{i,\gamma}(w) \, \tilde{p}_{i,\gamma}(w) \, \frac{\gamma}{w^2} \quad \sim_{LD} \quad \exp(-\tilde{I}_i(w)\gamma)$$

where

$$\tilde{I}_0(w) \ = \ \begin{cases} \dfrac{(c_0 w - 1) \ - \ \log(c_0 w)}{w} & \text{for } w \leq \mu_X \\[2ex] \dfrac{I(w) + \ (c_0 w - 1) \ - \ \log(c_0 w)}{w} & \text{for } w > \mu_X \end{cases} \tag{13}$$

and a similar expression defines $\tilde{I}_1(w)$. The minimum rate is

$$I_i^* \ = \ \inf_{w \geq 0} \ \tilde{I}_i(w), \tag{14}$$

and by Laplace's method we conclude that

$$P_{i,\gamma} \quad \sim_{LD} \quad \int_0^\infty \exp(-\tilde{I}_i(w)\gamma) \, dw \quad \sim_{LD} \quad \exp(-I_i^*\gamma). \tag{15}$$

In Figure 3 we have plotted the rate functions for the same example as in Figure 2. $\tilde{I}_0(w)$ and $\tilde{I}_1(w)$ are plotted as the bold solid lines. The thin solid line is $I(w)/w$ and the

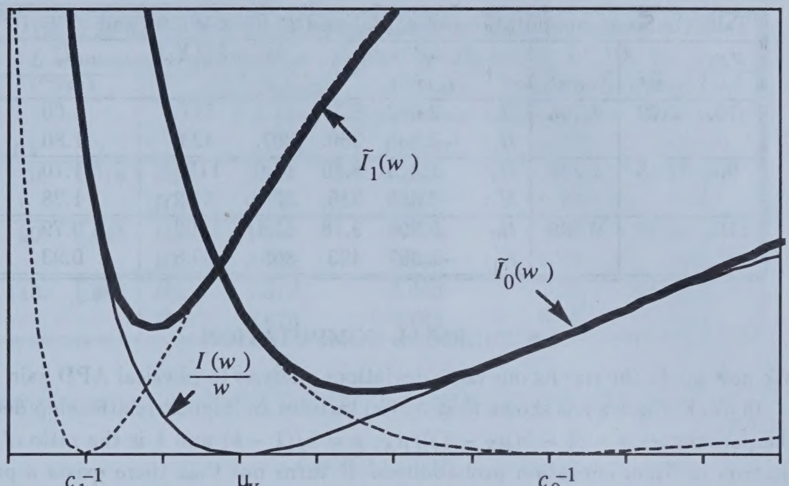


Figure 3: Plots of $\tilde{I}_0(w)$ and $\tilde{I}_1(w)$.

dashed lines are the functions $(c_i w - 1) - \log(c_i w)/w$ with $i = 0$ or 1 being clear from the minima at c_i^{-1}.

The next proposition characterizes the minimization in (14). It turns out that the rate functions $\tilde{I}_i(w)$ are not convex. Nonetheless, we have shown that there are well defined unique global minima w_i^* which can be identified as the solution of a nonlinear equation. The following theorem then gives our main result, a tight asymptotic expression for $P_{i,\gamma}$.

Proposition 4 *Assume the hypothesis of Proposition 1 and* $c_0 < \mu_X^{-1} < c_1$. *Then* $\tilde{I}_i(w)$ *has a unique minima* w_i^* *with* $\tilde{I}_i'(w) < 0$ *for all* $w < w_i^*$ *and* $\tilde{I}_i'(w) > 0$ *for all* $w > w_i^*$. *Furthermore,* w_i^* *can be identified as follows. First, determine the parameter* α_i^* *as the unique solution of the equation*

$$\Lambda_X'(\alpha_i^*) \quad = \quad (1 - c_G \alpha_i^*)\, c_i^{-1} \exp(-\Lambda_X(\alpha_i^*)). \tag{16}$$

Then $w_i^* = c_i^{-1} \exp(-\Lambda_X(\alpha_i^*))$ *and* $\alpha^*(w_i^*) = \alpha_i^*$. *Moreover, for* H_0 *we have* $w_0^* \in (\mu_X, c_0^{-1})$ *and* $\alpha_0^* > 0$, *and for* H_1 *we have* $w_1^* \in (c_1^{-1}, \mu_X)$ *and* $\alpha_1^* < 0$.

Theorem 1 *Assume the hypothesis of Proposition 1 and* $c_0 < \mu_X^{-1} < c_1$. *Then*

$$P_{i,\gamma} \quad \sim \quad \frac{C_i^*}{\sqrt{\gamma}} \exp(-I_i^* \gamma) \tag{17}$$

where

$$C_i^* \quad = \quad \sqrt{\frac{w_i^*}{2\pi (\alpha_i^*)^2 (\Lambda_X'(\alpha_i^*)^2 + \Lambda_X''(\alpha^*) + c_G w_i^*)}}, \tag{18}$$

and we can evaluate the rate as $I_i^* = \alpha_i^* - (\Lambda_X(\alpha_i^*) + c_G w_i^* (\alpha_i^*)^2/2)/w_i^*$.

Table 1: Some computations of α_i^*, w_i^* and I_i^* for $k = 0.04$ and $\sigma_G^2 = 10^6$.

μ_X	γ ($\times 10^4$)	$\bar{\alpha}$ ($\times 10^{-3}$)		α_i^* ($\times 10^{-3}$)	λ_i	λ_i^*	$E_i^*[X_k]$	I_i^* ($\times 10^{-3}$)
70.	1.02	2.106	H_0:	2.082	8.55	11.3	717.	1.60
			H_1:	-2.985	360.	307.	42.8	1.80
90.	1.45	1.324	H_0:	1.315	8.80	11.0	1190.	1.10
			H_1:	-2.386	385.	329.	51.2	1.28
110.	1.99	0.909	H_0:	0.905	9.18	11.1	1709.	0.79
			H_1:	-1.897	423.	365.	59.8	0.93

NUMERICAL COMPUTATION

We now apply the results our large deviations analysis to physical APD gain distributions. In [5], Personick has shown that $\Lambda_X(\alpha)$ satisfies an implicit relationship determined by the parameters $a = (1 - k(\mu_X - 1))/\mu_X$, $\rho = k/(1-k)$ and k is the ratio of the hole to electron collision-ionization probabilities. It turns out that there exists a parameter $\bar{\alpha} \in (0, \infty)$ such that $\Lambda_X(\alpha) \leq \infty$ for all $\alpha \leq \bar{\alpha}$ and $\Lambda_X(\alpha) = \infty$ for all $\alpha > \bar{\alpha}$. Nonetheless, it can be demonstrated from Personick's formula that $\Lambda_X(\alpha)$ satisfies the hypothesis of our Theorem 1. The avalanch gain probability mass function is also known. This probability mass function was rigorously derived by Mazo and Salz in [3], but it had actually been previously conjectured by McIntyre in [4] from some special case computations.

In addition to estimating the bit error probabilities using the asymptotic expression (17), we shall also obtain very accurate estimates using the Monte Carlo method known as *importance sampling*. Importance sampling has previously been applied to estimate bit error probabilities for APD receivers in [6], but the simulation designs proposed in these previous works have been ad hoc and have produced only modest improvement in computational efficiency. Here we propose a simulation distribution that is closely linked to large deviations theory using the results of [1]. Space does not allow a thorough exposition of this technique. Our simulation distribution for N is Poisson distribution with parameter $\lambda_i^* = \gamma/w_i^*$. Intuitively, we see that this distribution will emphasize samples near the critical value $n \approx \gamma/w_i^*$. The simulation distribution for the avalanch gain sequence is obtained from the large deviations twisting formula $f_{X,i}^*(x) = \exp(\alpha_i^* x - \Lambda_X(\alpha_i^*)) f_X(x)$. An analogous twisting formula is also used to bias the Gaussian noise in the importance sampling simulation.

Table 1 gives some sample numerical computations. The values of μ_X, γ, k and σ_G^2 are the same as those in Table II in Helstrom's paper [2]. The values of λ_i also corresponds to the entries in Helstrom's table. To indicate the effect of the "biasing" of the simulation distribution, Table 1 also plots the simulation distribution parameters λ_i^* and the baised mean avalanche gain $E_i^*[X_k]$ (which should be compared to the unbiased mean μ_X).

Table 2 presents bit error probability estimates for the same parameter values in Table 1. The figures in the "Saddlepoint Approximation" column were taken from Table II in Helstrom's paper [2]. The column labeled "Asymptotic Expression" is formula (17). The column labeled "Importance Sampling" gives importance sampling estimates obtained using the α_i^*-twisted distribution, and the "Standard Deviation" column provides empirical estimates of the importance sampling estimator's standard deviation obtained from the

Table 2: Bit error probability estimates for $k = 0.04$ and $\sigma_G^2 = 10^6$. L = number of simulations = 20,000 for H_0 and $L = 2,000$ for H_1.

μ_X	γ		Saddlepoint Approx. $(\times 10^{-9})$	Asymptotic Expression $(\times 10^{-9})$	Importance Sampling $(\times 10^{-9})$	Standard Deviation $(\times 10^{-11})$
70.	1.016	H_0:	1.162	1.229	1.15	5.1
		H_1:	0.835	0.857	0.92	4.7
90.	1.445	H_0:	1.250	1.358	1.23	6.3
		H_1:	0.732	0.748	0.73	3.7
110.	1.992	H_0:	1.319	1.435	1.33	7.3
		H_1:	0.670	0.684	0.65	3.5

sample variance estimator. Notice that the three estimates are quite close.

CONCLUSION

The problem formulation presented here is somewhat idealized. A primary performance limiting factor in many optical communications systems is intersymbol interference (ISI) which we have not considered here. In addition, most practical receivers replace the integrator in Figure 1 by an equalizing filter designed to minimize the ISI. The extension of our large deviations analysis to these important cases is the topic of ongoing research.

References

[1] J. A. Bucklew, P. Ney and J. S. Sadowsky (1990), "Monte Carlo simulation and large deviations theory for uniformly recurrent Markov chains," *J. Appl. Prob.*, vol. 27, pp. 44-59.

[2] C. W. Helstrom (1988), "Computing the performance of Optical Receivers with Avalandhe Diode Detectors," *IEEE Trans. on Commun.*, vol. 36, pp. 61-66.

[3] J. E. Mazo and J. Salz (1976), "On optical data communications via direct detection of light pulses," *Bell Syst. Tech. J.*, vol. 55, pp. 347-369.

[4] R. J. McIntyre (1972), "The distribution of gains in uniformly multiplying avalanche photodiodes: Theory," *IEEE Trans. Electron Devices*, vol. 19, pp. 703-712.

[5] S. D. Personick (1971), "Statistics of a general class of avalanche detectors with applications to optical communications," *Bell Syst. Tech. J.*, vol. 50, pp. 3075-3095.

[6] S. D. Personick, P. Balaban, J. H. Bobsin and P. R. Kumar (1977), "A detailed comparison of four approaches to the calculation of the sensitivity of optical fiber system receivers," *IEEE Trans. on Commun.*, vol. 25, pp. 541-548.

Performance Bounds for Coherent Lightwave BFSK with Hard-Decision Decoding

Yeheskel E. Dallal and Shlomo Shamai (Shitz)
Department of Electrical Engineering
Technion — Israel Institute of Technology
Haifa 32000, Israel

Abstract — A novel class of rigorous upper and lower bounds on error probability in coded binary frequency shift keying (BFSK) orthogonal communication schemes incorporating hard-decision decoding is presented. The bounds are applicable to assess the design and performance of the coherent lightwave heterodyned systems, wherein non-zero laser linewidths are evidenced. The effects of a Brownian carrier phase noise as well as an additive white Gaussian noise (AWGN) are addressed. Based on a power moment statistical characterization, the classical theory of Tchebycheff systems enables to determine the limiting values of a certain stochastic expectation involved, which corresponds to a bounded non-linear functional of a Brownian motion sample path. The analytic approach taken yields the tightest possible error bounds for the given moment knowledge covering arbitrary linewidths to information rate ratios. In general, the bounds obtained are remarkably tight and need only modest computational efforts. The developed theory is exemplified through examination of the capabilities of simple repetitions codes as well as the concatenated codes formed by combining the Hamming [7,4] code with repetitions codes. Insight into the selection of effective repetition–rates is pointed out. The results indicate the potential gain of coding for the suppression of the phase noise inevitable effects.

I. Introduction

Optical heterodyne detection has the potential for nearly ideal quantum limited performance [1], [2]. The exploitation of which has been limited by the poor coherence of semiconductor diode lasers. Due to the phase instability, often modeled by the Brownian motion, incoherent detection techniques, which are inherently less sensitive to phase noise, appear attractive [1]. Efficient uncoded modulation formats combined with quadratic detection have received extensive attention. A representative reference list is provided by [2]–[4] and references thereby.

As has been well recognized, coding provides efficient means to combat interference on a variety of channels among which are fading channels, partial band interference channels and multiple access interference channels. However, the merits of coding on the Brownian phase interference channel are not yet established neither widely accepted as for the above channels.

One theoretical reason for this is, perhaps, the fundamental difficulty to determine the exact statistics of non-linear functionals of the Brownian phase sample path, which rise when noisy phase signals are filtered. This statistical problem plays a key role in the error probability analysis of many of the proposed schemes. Various approaches have been adopted to overcome the above difficulty. Analytical approximation techniques and empirical models have been widely suggested and treated by many researchers [2], [5]–[7]. Importance sampling techniques were employed to simulate the envelope-statistics of a filtered noisy phase tone [8]. The numerical solution of the underlying Fokker–Planc equation is considered in [9]. In a recent study [4], [10] an upperbound on the error probability based on the use of the Chernoff bound technique is pursued. It yields a strict upperbound in a crosstalk limited regime that relies on a bivariate generating function of certain functionals of the phase sample path. These functionals that take on the interpretation of the in-band signal energy and the crosstalk energy and their moments are computable in closed form. The unknown generating function is bounded based on these available moments.

In this paper we consider the performance of the scheme depicted in Fig. 1, which consists of coded orthogonal BFSK operating over a noisy phase channel, where hard–quantization on each transmitted binary symbol is used. For a more detailed treatment of the receiver structure, see [10]. This suboptimal demodulator has been widely accepted for coherent lightwave communication [1]-[4], [6]-[7], [10]. The resulting discrete time memoryless binary–symmetric channel (BSC) accounts for the BFSK modulator, the noisy phase channel, the time–diversity reception which is accomplished by means of prefiltering followed by square–law–combining and for the hard decisions. This BSC is completely characterized by its crossover probability, which is typically large and ranges from 10^{-1} to 10^{-4} in coded systems. In contrast to uncoded systems [10], here relatively low diversity levels are of interest since much of the required redundancy is provided by means of powerful coding rather than achieved by the embedded diversity combining technique. In evaluating the performance of a hard-decision decoding algorithm on the considered BSC, it is of primary concern to derive firm bounds on the crossover probability, which are tight at the realm of high probability values and low diversity levels. Though the Chernoff bound is applicable, it yields rather loose results for these conditions. Roughly speaking, it is useful primarily in situations involving the far tails of the distribution function of the sums of sufficiently many independent random variables, which is evaluated at points largely deviated from the mean, as is the case for the uncoded diversity combining coherent lightwave communication [10], [11].

In the sequel we shall take a different approach, which utilizes a power moment characterization of certain non–linear bounded functionals of the Brownian motion sample path to provide strict effective upper and lower bounds on the BSC crossover probability. This new class of bounds is useful to assess the design and the performance of coded coherent lightwave communication systems employing quadratic detection with hard–decoding and yields the tightest possible bounds on the crossover probability implied by the available moment knowledge. The corresponding results are obtained in algebraic closed form expressions and facilitate, therefore, a convenient and a straightforward application.

II. Model Formulation

Consider an orthogonal binary frequency shift keying (FSK) signaling in the presence of both additive white Gaussian noise (AWGN) and a Brownian phase impairment. The complex representation of the observation for the null symbol is

$$V_t = S_t + N_t \qquad t \in [0,T)\,, \tag{1}$$

where T stands for the binary symbol duration, $\{N_t\,, t \in \mathbb{R}\}$ represents an AWGN with one-sided power spectral density N_o and $\{S_t,\ t \in [0,T)\}$ corresponds to the noisy phase information bearing signal

$$S_t = \sqrt{2P}\ e^{j(\omega_m t + \phi_t + \theta)} \qquad m=1\,,2. \qquad t \in [0\,,T)\,. \tag{2}$$

The information is conveyed by the two frequencies $\{\omega_m\}_{m=1}^{2}$, P is the signal power, $\{\phi_t\,, t \in \mathbb{R}\}$ is a phase noise process defined by

$$\phi_t = \sqrt{2\pi B_l}\ \mathbf{W}_t \qquad t \in \mathbb{R}\ , \tag{3}$$

where $\{\mathbf{W}_t\,, t \in \mathbb{R}\}$ is the standard Brownian Motion [12] and the involved diffusion coefficient implies that the resulting 3db linewidth is B_l. Finally θ is a random variable uniformly distributed over $[-\pi\,,\pi)$, which accounts also for an arbitrary time shift of the phase process.

The binary FSK demodulator considered here utilizes an L-fold time diversity with soft combining as depicted in Fig. 2. The overall symbol duration T is divided to L disjoint time chips of length T_c, thus:

$$T_c = T/L\ . \tag{4}$$

The chip inner products

$$v_{ml} = \int_0^{T_c} V_{t+lT_c} e^{-j\omega_m (t+lT_c)} g^*(t)\,dt \quad m=1,2 \quad l=0,1,\ldots,L-1\,, \tag{5}$$

of the observation $\{ V_t , t \in [0,T) \}$ are computed each symbol period T and over each hypothesized frequency $\{ \omega_m \}_{m=1}^{2}$. Here $g(t)$ is a unit norm function in $L^2[0,T_c]$, namely $\|g\|^2 = 1$. Notice that (5) renders a linear prefiltering of the observation with a complex baseband equivalent impulse response $g^*(T_c - t)$. The purpose of the filtering is to limit the additive noise bandwidth while retaining most of the energy in the information carrying signal.

The next step in processing the signal is square envelope detection. Given m, it is straightforward to show that the L quadratic norms $\{ |v_{ml}|^2 \}_{l=0}^{L-1}$ are identically distributed and statistically independent (i.i.d.) [10]. This observation suggests the inclusion of a sort of combining to exploit the scheme diversity for mitigating some of the phase interference detrimental effects [4], [10], [11]. This can be simply accomplished as follows: for each binary symbol, a hard quantization is made by the demodulator on the sign of the decision variable q set by the difference of the two quadratic forms,

$$q = \sum_{l=0}^{L-1} |v_{2l}|^2 - \sum_{l=0}^{L-1} |v_{1l}|^2 . \tag{6}$$

Likewise, estimates of the symbol energy in each frequency band are compared to determine the transmitted binary symbol. The resulting binary channel is both memoryless and symmetric (BSC) and is characterized solely by p the crossover probability

$$p = \text{Prob}(q < 0 \mid \omega_2 \text{ was sent})\ , \tag{7}$$

wherein due to symmetry no loss in generality results by letting ω_2 be the transmitted frequency. In the next section we shall undertake the derivation of a class of upper and lower bounds on p, which turns useful for coded systems operating over the BSC.

III. Crossover Probability Bounds

Given the phase sample path $\{ \phi_t\,,\ t \in [0,T] \}$, the conditional probability is just the probability that the difference of two independent chi-squared variables of 2L degrees of freedom (by virtue of the orthogonality assumption one variable is non–central and the other is central) is less than zero and is given by [13],

$$p(s_o, s_1, \cdots, s_{L-1}) = \Omega(z, L) \Big|_{z = \frac{1}{2} \frac{PT_c}{N_o} \sum_{l=0}^{L-1} s_l} \tag{8}$$

where the function $\{ \Omega(z, L),\ \ z \geq 0 \}$ is defined by

$$\Omega(z, L) \triangleq \tfrac{1}{2} e^{-z} \sum_{i=0}^{L-1} C_i(L) \frac{z^k}{i!} \qquad z \geq 0\ , \tag{9}$$

and the summation in (9) is a polynomial in z of the $L-1$th degree with the coefficients

$$C_i(L) = \frac{1}{(L-1+i)!} \sum_{j=i}^{L-1} \frac{(L-1+j)!}{(j-i)!\, 2^{L-1+j}} \qquad i=0,1,\cdots,L-1\,. \tag{10}$$

The notation $p(s_o, s_1, \cdots, s_{L-1})$ is used to emphasize the explicit dependency on the phase sample path functionals $\{ s_l \}_{l=0}^{L-1}$ defined by

$$s_l = \frac{1}{T_c} \left| \int_0^{T_c} e^{j\phi_{t+lT_c}} g^*(t)\,dt \right|^2 \qquad l=0,1,\cdots,L-1\,. \tag{11}$$

Lemma 1 outlines some useful properties of the foregoing functionals [13], whose exact statistics is unknown. For a detailed exposition the reader is referred to [10].

Lemma 1: Let $\{ s_l \}_{l=0}^{L-1}$ be the functionals defined in (10), whereas $\{ \phi_t , t \in \mathbf{R} \}$ is a Brownian phase (3). Then:

(L1.A) The $\{ s_l \}_{l=0}^{L-1}$ are i.i.d. .

(L1.B) s_o is a non-negative bounded functional of $\{ \phi_t , t \in [0, T_c] \}$, such that $0 \leq s_o \leq 1$.

(L1.C) The power moments of s_o are tractable though the corresponding distribution is analytically unknown. Analytical techniques to efficiently obtain these moments are developed in [10]. The derivation involves as well some straightforward manipulation using Itô calculus. Let ε_k be the k'th power moment, namely

$$\varepsilon_k \triangleq E \{ s_o^k \} \tag{12}$$

□

Based on the above incomplete statistical characterization we now seek for the tightest possible upper and lower bounds on the expectation

$$p = E \{ p(s_o, s_1, \ldots, s_{L-1}) \} = \int_0^1 \ldots \int_0^1 \Omega \left[\frac{1}{2} \frac{PT_c}{N_o} \sum_{l=0}^{L-1} z_l , L \right] dF(z_o) \ldots dF(z_{L-1}) \ . \tag{13}$$

The classical theory of Tchebycheff systems, which plays a prominent role in studying the extremal values of Stieltjes integrals with respect to a distribution specified via its moments, can be utilized here to derive the desired bounds on p. Indeed the extremal distributions $\overline{F}$ and $\underline{F}$, which achieve the bounds are readily solved if $\Omega \left[\frac{1}{2} \frac{PT_c}{N_o} (z + \delta), L \right]$, $\delta \in [0, L-1]$ is restricted to form a weak Tchebycheff system with the classical powers $\{ 1, z, \cdots, z^K \}$ on the closed interval [0, 1]. The solution of the extremal distributions and the corresponding bounds is furnished by theorem 1. The details are provided in [13].

Definition 1: Let $A(z, K, L)$ be the $L-1$th order polynomial defined by

$$A(z, K, L) \triangleq \sum_{l=0}^{L-1} \alpha_l(K, L) \frac{z^l}{l!} \ , \tag{14a}$$

$$\text{if } L \leq K+2 \text{ then } \alpha_l(K, L) = \sum_{i=l}^{L-1} \lambda_i(l, K, L) \qquad 0 \leq l \leq L-1 \ .$$

$$\text{if } L > K+2 \text{ then } \alpha_l(K, L) = \begin{cases} \displaystyle\sum_{i=l}^{L-1} \lambda_i(l, K, L) & L-K-2 \leq l \leq L-1 \\ \displaystyle\sum_{i=l}^{K+1+l} \lambda_i(l, K, L) & 0 \leq l \leq L-K-3 \end{cases} \tag{14b}$$

$$\lambda_i(l, K, L) = C_i(L) \begin{bmatrix} K+1 \\ i-l \end{bmatrix} (-1)^{K+1+l-i} \qquad l \leq i \leq L-1 \tag{14c}$$

In (14c) the $C_i(L)$ are given by (10) and $\left[: \right]$ denotes the binomial coefficient.

Definition 2: We distinguish the following cases regarding the locus of the real zeros of the polynomial $A(z, K, L)$:

(D2.A) The polynomial $A(z, K, L)$ has no nodal zeros in the real semi-infinite open interval $(0, \infty)$ (A zero is nodal if the polynomial changes sign in its neighbourhood).

(D2.B) The polynomial $A(z, K, L)$ has nodal zeros in $(0, \infty)$. Denote the smallest nodal zero by z^*.

Definition 3: The principal representations admitted by the power moment sequence $\{ \varepsilon_k \}_{k=1}^{K}$ are uniquely defined as the two discrete distributions of index $K+1$ supporting the given moments $\{ \varepsilon_k \}_{k=1}^{K}$ on [0, 1] (The index of a discrete distribution on [0, 1] is that number obtained by counting interior roots, i.e. interior points of probability concentration, as two and roots at either end points as one). A principal representation is further designated by the term upper if it involves endpoint 1, otherwise it is called a lower principal representation.

Theorem 1: Let $\Omega(z, L)$ be given by (9) and its corresponding polynomial $A(z, K, L)$ by (14a)–(14c). Suppose that for case (D2.A) K is odd and that for case (D2.B) the first nontrivial coefficient of $A(z, K, L)$ is positive and that the symbol energy to noise density ratio satisfies $z^* \geq \frac{LPT_c}{2N_o} \geq 0$. Then:

(T1.A) The external distributions $\bar{F}$ and $\underline{F}$, respectively attaining the tightest possible upper and lower bounds on p (13) for the incomplete statistical characterization provided by Lemma 1 utilizing the K moments $\{ \varepsilon_k \}_{k=1}^{K}$ (12) are given by the, so called, principal representations admitted by the sequence $\{ \varepsilon_k \}_{k=1}^{K}$ on [0, 1].

(T1.B) The upperbound is readily given by the $\bar{\nu}$-fold summation

$$p \leq \sum_{\substack{j_1, j_2, \cdots, j_{\bar{\nu}} \\ j_1 + j_2 + \cdots + j_{\bar{\nu}} = L}} \frac{L!}{j_1! j_2! \cdots j_{\bar{\nu}}!} \bar{\rho}_1^{j_1} \bar{\rho}_2^{j_2} \cdots \bar{\rho}_{\bar{\nu}}^{j_{\bar{\nu}}} \, \Omega\left[\frac{PT_c}{2N_o} (j_1 \bar{\xi}_1 + j_2 \bar{\xi}_2 \cdots + j_{\bar{\nu}} \bar{\xi}_{\bar{\nu}}), L \right], \tag{15}$$

where $\{ \bar{\rho}_i, \bar{\xi}_i \}_1^{\bar{\nu}}$ are respectively the roots and probability weights associated with the upper principal representation of the moments $\{ \varepsilon_k \}_{k=1}^{K}$.

(T1.C) The lowerbound is readily evaluated by the $\underline{\nu}$-fold summation

$$p \geq \sum_{\substack{j_1, j_2, \cdots j_{\underline{\nu}} \\ j_1 + j_2 + \cdots + j_{\underline{\nu}} = L}} \frac{L!}{j_1! j_2! \cdots j_{\underline{\nu}}!} \underline{\rho}_1^{j_1} \underline{\rho}_2^{j_2} \cdots \underline{\rho}_{\underline{\nu}}^{j_{\underline{\nu}}} \, \Omega\left[\frac{PT_c}{2N_o} (j_1 \underline{\xi}_1 + j_2 \underline{\xi}_2 \cdots + j_{\underline{\nu}} \underline{\xi}_{\underline{\nu}}), L \right], \tag{16}$$

where $\{ \underline{\rho}_i, \underline{\xi}_i \}_1^{\underline{\nu}}$ are respectively the roots and weights associated with the lower principal representation of the moments $\{ \varepsilon_k \}_{k=1}^{K}$.

□

Remark 1: For case (D2.A) with an even K and for case (D2.B) where a negative coefficient results, the role of the upper and lower principal representations is interchanged. Thus RHS of equation (15) yields a lowerbound whereas (16) constitutes an upperbound.

Remark 2: For the classical power moments treated in this paper, the solution of the interior roots of the principal representations can be obtained in terms of certain polynomials having simple zeros at these points. These polynomials are constructed according to the assigned moments $\{ \varepsilon_k \}_{k=1}^{K}$ and do not depend on the values of the corresponding probability weights [10], [11], [13].

IV. Coding Performance

Here we illustrate the application of the developed theory to the design and the performance analysis of BFSK coding systems, which utilize hard-decision decoding subjected to both a Brownian phase interference and an AWGN. A rectangular baseband impulse response is assumed throughout, i.e. the prefilter $g(t)$ (5) is $g(t) = 1/\sqrt{T_c}$. It is noteworthy that though this particular choice constitutes a matched filter for the phase noise free signal, it is suboptimal in the presence of the Brownian phase noise [14]. The closed form expressions for the power moments, utilized in this paper, are provided in [13]. In the sequel we study the performance of repetition codes and repeated Hamming codes.

IV.A. *Repetition codes*

Repetition codes form a trivial class of codes. A binary repetition code $\mathcal{R}(J)$, is a $[J\,,1\,,J]$ linear block code, which results by subdividing each information bit into J symbols. Odd values of J yield a perfect code, which is strictly capable of correcting all error patterns of weight $\frac{J-1}{2}$ or less. Hence the information bit error probability is expressed by

$$P_b = \sum_{k=\frac{J-1}{2}+1}^{J} \begin{bmatrix} J \\ k \end{bmatrix} p^k (1-p)^{J-k} \quad , \tag{17}$$

where p is the crossover probability. Now substituting (15) and (16) in (17) yields the desired upper and lower bounds on P_b. The flexibility in chosing the repetition rate J enables an optimal matching of the $\mathcal{R}(J)$ code rate to the channel. The existence of optimum rates is a common feature of coded incoherent schemes [15] and presents a tradeoff between lack of redundancy at low repetition rates and incoherent decoding loss at high repetition rates.

Performance bounds on P_b for optimal repetition codes with $L=1$ are provided in Fig. 3. Each plot admits a specified linewidth–to–information–bit ratio B_l/R and presents a bound on P_b which is optimized versus J at a given bit energy–to–noise density ratio E_b/N_o. The tightness of the resulting bounds, which are based on the availability of only three power moments (i.e. K=3), is apparently adequate for the considered broad range of bit error probabilities and system parameters. At $B_l/R = 10^{-1/2}$ and $P_b \sim 10^{-9}$ the bounds diverge in roughly 0.3dB and their tightness typically improves as either B_l/R or P_b are increased.

As previously mentioned, the noisy phase BFSK channel with quadratic detection is directly converted into a memoryless channel. Unlike coherent or differential detection when used over the Brownian phase interference channel [11], [15], here the necessity for interleaving techniques is entirely eliminated. Consequently any $\mathcal{R}(J)$ code poses no encoding requirements on the transmitter and is essentially a form of time diversity reception often referred to as a majority–count combining [16]. The receiver divides each bit period into J chips, whereas hard decisions are made in each chip and the decoded bit is set equal to that value which appears in a majority of chip decisions. It is conceivable that soft combining techniques would show some performance gain relative to the majority-count combiner. Indeed, comparing the performance of optimal repetition codes (L=1) with the results of BFSK with an optimally designed L-fold square-law combining [4], [10] we find a typical loss of 1.6dB at $P_b \sim 10^{-9}$ and $B_L/R \sim 1$. This moderate loss might be tolerated in some heterodyned lightwave systems due to the attendant simplification in implementing the majority-count combiner.

IV.B. *A repeated Hamming code*

The choice of actual code rates might be limited due to implementation considerations. If the time redundancy in a given code is not sufficient J–fold repetitions of the encoded bits are useful to create an overall code with a sufficiently low rate. Here we investigate the performance of a repeated Hamming code. Binary Hamming codes form a simple nontrivial class of single-error-correcting codes, which are easy to encode and decode. The $\mathcal{H}_3$ code is a linear block code of length 7, dimension 4 and minimum distance 3, i.e. $\mathcal{H}_3$ is a $[7\,,4\,,3]$ code which is unique up to equivalence. The weight enumerator for $\mathcal{H}_3$ is $1+7X^3+7X^4+X^7$ [17]. J-fold repetitions of the $\mathcal{H}_3$ codewords create a $[7J\,,4\,,3J]$ binary code denoted $\mathcal{H}_3(J)$. Obviously, $\mathcal{H}_3(J)$ has the weight enumerator given by the polynomial $1+7X^{3J}+7X^{4J}+X^{7J}$. Though the original code $\mathcal{H}_3$ is perfect it can be verified that the corresponding repeated code $\mathcal{H}_3(J)$ is no longer perfect.

For illustration consider a bounded distance decoding for the $\{\,\mathcal{H}_3(J)\,\}_J$ codes, which are derived from a systematic $\mathcal{H}_3$ code. The decoder corrects $t = \left\lfloor \frac{3J-1}{2} \right\rfloor$ or fewer errors in a received codeword. Undetected errors result as the received word lies within a decoding sphere surrounding incorrect codewords. With non perfect codes, i.e. $J > 1$, the decoder fails to decode whenever the received word lies in the interstices outside the decoding spheres. In this case an erasing decoder discards the received word. For an erasing decoder with the $\{ \mathcal{H}_3(J) \}$ codes, it can be shown that the error rate of the decoded information bits equals exactly the decoded bit error rate [13]. The assertion is strictly valid for a repeated systematic $\mathcal{H}_3$ code but may be violated for other equivalent $\mathcal{H}_3$ codes. The above observation implies that the information bit error rate, denoted Pib, depends on the code structure solely in terms of its code weight distribution and yields [18]

$$Pib = \sum_{m = 3J - \left\lfloor \frac{3J-1}{2} \right\rfloor}^{7J} p^m (1-p)^{7J-m} \sum_{l = \max\left[3J, m - \left\lfloor \frac{3J-1}{2} \right\rfloor\right]}^{\min\left[7J, m + \left\lfloor \frac{3J-1}{2} \right\rfloor\right]} \frac{l}{7J} \mathbf{A}_l(J) \mathcal{N}(l, m) \tag{18}$$

$$\mathcal{N}(l, m) = \sum_{s=0}^{\left\lfloor \frac{3J-1}{2} \right\rfloor} \sum_{\substack{j=0 \\ 2j = s+l-m}}^{7J} \begin{bmatrix} 7J - l \\ j + m - l \end{bmatrix} \begin{bmatrix} l \\ j \end{bmatrix} , \tag{19}$$

where the $\{ \mathbf{A}_l(J) \}_{l=3J}^{7J}$ denote the code weight coefficients of $\mathcal{H}_3(J)$. The convention, $\begin{bmatrix} m \\ l \end{bmatrix} = 0$ if l is negative or larger than m, is used for the binomial coefficients. A corresponding expression for the probability of decoding failure, denoted P_{df}, can be obtained via [19]. In table 1 we provide the performance bounds (L=1) for optimally repeated $\mathcal{H}_3(J)$ codes assuming $P_{df} \leq 10^{-3}$. The performance measure is the minimal E_b / N_o required to achieve $Pib \leq 10^{-9}$. A variety of phase noise levels are exemplified based on the availability of three power moments, i.e. $K = 3$. The results, presented, are remarkably tight and reveal a fairly robust performance versus the linewidth to bit rate ratio.

V. Discussion

Quadratically detected coded BFSK with hard–decoding is examined for coherent lightwave communication. Tight performance bounds, which utilize the availability of the power moments of the involved bounded non–linear functional of the Brownian phase sample path, were presented and their effectiveness was demonstrated. Though the required statistics is a mathematically intractable problem, the approach taken here lends itself to a rigorous analysis, which optimally utilizes the available incomplete statistical knowledge to achieve the tightest possible bounds. The choice of quadratic detection with hard-decision decoding was shown to feature a highly robust behavior against the phase interference as well as implementation advantages. It was illustrated that power efficient coded systems can be designed to operate at substantial phase noise levels. A corresponding scheme which is based on a binary DPSK rather than BFSK has been suggested in [20] and analyzed using the simplifying assumptions of [2]. The comparison of these results with the results for the coded BFSK scheme considered here, indicates a significant bandwidth reduction with BFSK as well as power advantages at excessive levels of phase noise, which along the relative simplicity makes the coded BFSK systems attractive for hetrodyned optical links.

References

[1] R.A. Linke, "Optical Heterodyne Communications Systems", IEEE Commun. Magazine, pp. 36–41, Oct. 1989.

[2] J. Salz, "Coherent Lightwave Communication", AT & T Tech. Jour. Vol. 64, No. 10, pp. 2153–2209, Dec. 1985.

[3] G.J. Foschini, L.J. Greenstein and G. Vannucci, "Non-coherent Detection of Coherent Lightwave Signals Corrupted by Phase Noise", IEEE Trans. on Commun., Vol. 36, No. 3, pp. 306–314, March 1988.

[4] Y.E. Dallal and S. Shamai (Shitz), "On the Error Probability of Noisy Phase Channels Using MFSK", The Sixteenth Conference of IEEE in Israel, Program and Papers 89–CH 2723–5, Paper 2.11, March 1989.

[5] L. Jeromin, V.W.S. chan, "M-ary FSK Performance for Coherent Optical Communication Systems Using Semiconductor Lasers", IEEE Trans. on Commun., Vol. 34, pp. 375–381, April 1986.

[6] I. Garrett and G. Jacobsen, "Theoretical Analysis of Heterodyne Optical Receivers for Transmission Systems Using (Semiconductor) Lasers with Non-Negligible Linewidth", Jour. of Lightwave Tech., Vol. LT-4, No. 3, pp. 323–334, March 1986.

[7] G.J. Foschini, G. Vannucci and L.J. Greenstein, "Envelope Statistics for Filtered Optical Signals Corrupted by Phase Noise", IEEE Trans. on Commun., Vol. 37, No. 12, pp. 1293–1302, Dec. 1989.

[8] G.J. Foschini and G. Vannucci, "Characterizing Filtered Light Waves Corrupted by Phase Noise", IEEE Trans. on Inform. Theory, Vol. IF-34, No. 6, pp. 1437-1448, Nov. 1988.

[9] I. Garrett et al., "Impact of Phase Noise in Weakly Coherent Systems: A New and Accurate Approach", J. of Lightwave Tech., Vol. 8, No. 3, pp. 329–337, March 1990.

[10] Y.E. Dallal and S. Shamai (Shitz), "An Upper Bound on the Error Probability of Quadratic Detection in Noisy Phase Channels", to appear in IEEE Trans. on Commun.

[11] Y.E. Dallal and S. Shamai (Shitz), "Coherent Lightwave DPSK with Time–Diversity", Proceedings of the 1990 Bilkent International Conference on New Trends in Commun., Control and Signal Processing, Vol. I, pp. 474–480, Ankara, Turkey, July 1990.

[12] E. Wong and B. Hajek, Stochastic Processes in Engineering Systems, Springer-Verlag Inc., New York, 1977.

[13] Y.E. Dallal and S. Shamai (Shitz), "Performance Bounds for Noisy Phase BFSK with Hard-Decision", Dept. of Electrical Eng., Technion-Israel Institute of Technology, Haifa, Israel, EE. Pub. No. 740, April 1990.

[14] Y. Dallal and S. Shamai (Shitz), "Asymptotic Behavior of MFSK in Noisy Phase channels", International Symposium of Inform. Theory (ISIT90), Session WA1, Abstract of Papers, p.90, San Diego, CA., January 14-19, 1990.

[15] Y.E. Dallal and S. Shamai (Shitz), "Concatenated Coding for Coherent Lightwave DPSK", to be presented in the 1990 International Symposium on Information Theory and Its Applications (ISITA'90), Hawaii, Nov. 1990.

[16] R.R. Reed and J.N. Pierce, "Comparison of Square-Law and Majority-Count Diversity Combiners", IEEE Trans. on Commun. Tech., Vol. COM-12, p. 217, Dec. 1964.

[17] F.J. Mac Williams and N.J.A. Sloan, The Theory of Error–Correcting Codes, North–Holland, New York, 1977.

[18] D. Torrieri, "Information-Bit, Information-symbol and Decoded-Symbol Error Rates for Linear Block Codes", IEEE Trans. on Commun., Vol. 36, No. 5, pp. 613-617, May 1988.

[19] R.E. Blahut, Theory and Practice of Error control Codes, Addison-Wesley, Reading, Mass., 1983.

[20] J.E. Mazo, "On Binary Differential Detection for Coherent Lightwave Communication", AT & T. Tech. J., Vol. 64, No. 10, Dec. 1985.

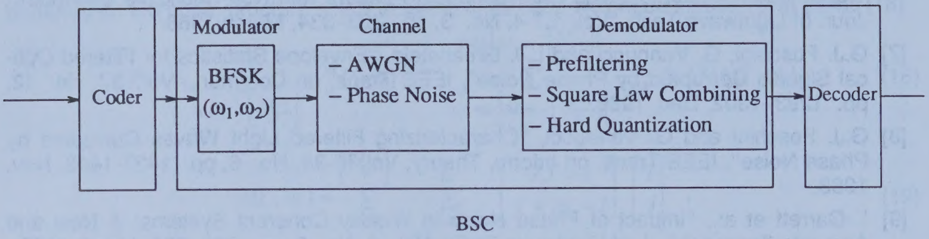

Coder
Modulator
BFSK
(ω1,ω2)
Channel
– AWGN
– Phase Noise
Demodulator
– Prefiltering
– Square Law Combining
– Hard Quantization
Decoder
BSC

Figure 1: System overview

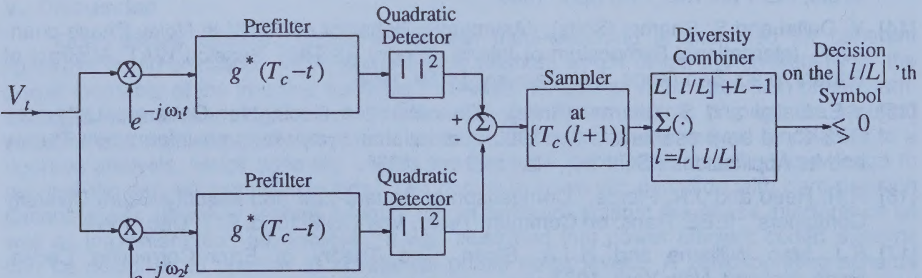

V_t
$e^{-j\omega_1 t}$
$e^{-j\omega_2 t}$
Prefilter
$g^*(T_c-t)$
Prefilter
$g^*(T_c-t)$
Quadratic Detector
$| \, |^2$
Quadratic Detector
$| \, |^2$
+
-
Σ
Sampler
at
$\{T_c(l+1)\}$
Diversity Combiner
$L\lfloor l/L \rfloor + L - 1$
$\Sigma(\cdot)$
$l = L\lfloor l/L \rfloor$
Decision on the $\lfloor l/L \rfloor$ 'th Symbol
$\lessgtr 0$

Figure 2: The complex equivalent form of the postulated demodulator

	B_l/R	0.01	0.02	0.05	0.1	0.2	0.5	1
Upper	E_b/N_o db	15.8	15.8	15.9	15.9	16.2	17.0	17.5
Bound	J	1	2	2	2	2	4	4
Lower	E_b/N_o db	15.8	15.8	15.9	15.9	16.1	16.7	17.4
Bound	J	1	1	2	2	2	2	4

Table 1: Bounds on the minimal E_b/N_o required to achieve the information bit-error rates $Pib \leq 10^{-9}$ for optimally repeated $\mathcal{H}_3(J)$ codes assuming erasing decoding with $P_{df} < 10^{-3}$ (K=3).

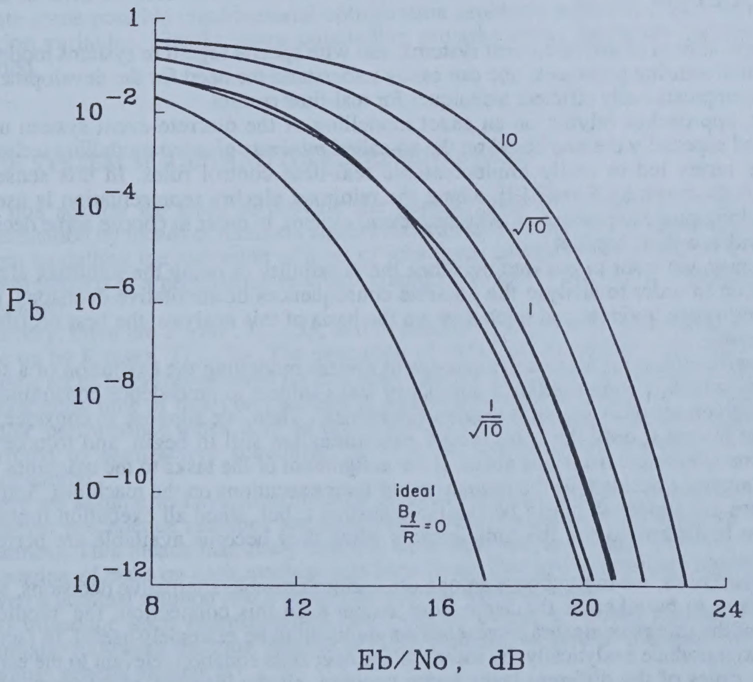


Figure 3: Performance bounds for optimal repetition codes showing the information bit error probability versus E_b/N_o at various B_l/R (K=3).

ALTERNATIVE ASSIGNMENT AND SEQUENCING SELECTION IN A DETERMINISTIC DISCRETE-EVENT SYSTEM BY MEANS OF THE MINIMAX ALGEBRA APPROACH

M. Aicardi, A. Di Febbraro, R. Minciardi

Department of Communications, Computer and Systems Science
University of Genoa
Via all'Opera Pia 11a - 16145 Genoa - Italy
Tel. 39-10-3532983 Fax 39-10-3532948

ABSTRACT

The paper presents the possibility of using the minimax algebra technique to analyze and optimize the performances of a certain class of discrete-event systems, where specific choices about the assignment and the sequencing of tasks are to be made. It will be shown how to determine a state equation of the discrete-event system able to represent every considered decisional alternative. Then, by using the minimax algebra resolvent matrix, it is possible to obtain direct dependences of certain performance indexes on the decisional variables under concern.

1. INTRODUCTION

In the general field of discrete-event systems, and with special regard to systems modelling discrete manufacturing processes, one can easily understand the need for the development of sound and computationally efficient techniques for real-time control.

Actually, approaches relying on an exact modelling of the discrete-event system under concern (and especially the one based on the so-called *minimax algebra* modelling technique [1,2]) have rarely led to really implementable real-time control rules. In this sense, an exception is the work by Engell [3], where the minimax algebra representation is used to foresee the long-range behaviour of a discrete-event system, in order to choose some decision variables within a short horizon.

In this paper, we want to put into evidence the possibility of using the minimax algebra representation in order to analyze the possible consequences of alternative decisions, even within a long-range horizon, and to choose, on the basis of this analysis, the best decision at the present time .

More specifically, we refer to a discrete-event system modelling the execution of a set of jobs, each of which is composed by a number of tasks subject to precedence constraints, by means of a given set of available resources (machines). Then, we suppose to consider, at a certain time instant t, only the tasks whose executions are still to begin, and to take into account some alternative decisions about: i) the assignment of the tasks to the machines and, when assignment is decided, ii) the sequencing of their executions on the machines. Some of the machines are supposed not to be available at time t, but, since all execution times are supposed to be deterministic, the time instants when they become available are perfectly known.

In such conditions, we must choose among the various possible alternative decisions, some of which have to be taken at the same time instant t. In this connection, the 'predictive capability' of the minimax algebra representation turns out to be extremely useful. In fact, we prove able to introduce analytically, by means of a proper state equation relevant to the earliest completion times of the different tasks under concern, all the 'degrees of freedom' of the system, i.e., to take into account all the possible alternative decisions that could be taken about the considered set of tasks.

In other words, we are able to perform, with respect to the variables representing the earliest completion times, a sort of 'perturbation analysis', somehow in the same sense as in [4], but taking into account at the same time, and in an exact way, several possible perturbations to a 'nominal solution'.

Namely, we can see that it is possible to make it explicit and analyze the dependence of the earliest completion times on the decision variables under concern, which are relevant to alternative assignment or sequencing. This fact, in turn, makes it possible to establish an analytic dependence of a suitable performance criterion, based on the earliest completion times only, on the same decision variables. Clearly, a complex optimization problem may arise, in this case, if the number of the decision variables is rather large. However, the minimax algebra 'resolution' of a discrete-event system allows to reduce the combinatorial optimization problem to its simplest form, since it can provide an analytic expression of the considered performance criterion which can be simplified by neglecting those decision variables which turn out not to have any influence on the considered performance index.

The possibility of using an analytic expression of the considered performance criterion as a function of the decision variables, and the expected decrease in the complexity of the resulting optimization problem can be clearly of great advantage in a real-time environment. In this case, one can think of using even suboptimal solutions of the 'reduced' combinatorial problem, so as to apply only the decisions which are to be taken at the present moment.Then, whenever a new decision has to be taken (due to the occurrence of some event), a new optimization problem will be built up, just in the same way, with the possible use of updated information (e.g., regarding the 'arrival' of new tasks to be executed).

The paper will be organized as follows. In the second section, the model of the considered discrete-event system will be presented, whereas in the third section the way to represent alternative possibilities in the assignment and sequencing of tasks will be detailed. In the fourth section, the predictive use of the minimax algebra representation will be used in order to state some possible combinatorial optimization problems with respect to the considered decision variables. Finally, some concluding remarks about the whole approach will be reported.

2. THE SYSTEM MODEL AND ITS REPRESENTATION

In this section we want to provide the general model of our system, as well as its representation by means of minimax algebra formalism. We consider a discrete-event dynamic system modelling the execution of a set of jobs (each of which composed by a fixed set of tasks organized in a precedence-relation graph) on a certain number of available resources (machines).

Namely, there are Z jobs , $J_1,...,J_Z$, and N machines, $M_1,...,M_N$. The whole of the jobs is made up by R tasks, $T_1,...,T_R$. The execution of each task T_i may be accomplished by one among a set of machines allowed for that task. Execution times are deterministic (t_{ij} represents the execution time of task i on machine j), and no preemption is allowed. We assume that a 'regular' performance index is to be optimized [5], which implies that our attention can be restricted to semi-active schedules, so that the optimization problem is completely combinatorial.

Here we assume to have determined a feasible nominal solution (a solution is feasible whenever precedence relations are satisfied and task executions are assigned to allowed machines). This means that every task has been assigned to a single machine and that the sequencing of tasks on each machine has been fixed. The use of minimax algebra allows to convert a nominal solution into a simple state-space representation.

To introduce such representation, we need to define a certain number of matrices, essentially following the same lines as in [1,3]. Namely, we define a boolean matrix A (R x R) whose generic element is

$$a_{ij} = \begin{cases} 1 & \text{if task } T_j \text{ is one of the immediate predecessors} \\ & \text{(in some precedence-relation graph) of task } T_i \\ 0 & \text{otherwise} \end{cases}$$

Secondly, we define a matrix F (R x R) whose generic element is

$$f_{ij} = \begin{cases} t_{ik} & \text{if task } T_j \text{ is one of the immediate predecessors} \\ & \text{of task } T_i \text{ in some precedence-relation graph} \\ & \text{and/or in the sequence of executions} \\ & \text{assigned to machine } M_k \\ -\infty & \text{otherwise} \end{cases}$$

and a matrix G(R x N) whose generic element is

$$g_{ik} = \begin{cases} t_{ik} & \text{if task } T_i \text{ is assigned to machine } M_k \\ -\infty & \text{otherwise} \end{cases}$$

In order to extend the generality of our approach (and, more specifically, to include the possibility of considering real-time scheduling), we explicitly allow the possibility of exogenous starting conditions for tasks and machines. Namely, let $\underline{s}$ an L-dimensional vector of starting conditions for tasks (corresponding to the fact that some predecessor tasks are presently under execution), and $\underline{m}$ an N-dimensional vector of starting conditions for machines (corresponding to the fact that some machines are presently working). Moreover, let matrix B(R x L) defined so that

$$b_{ij} = \begin{cases} t_{ik} & \text{if task } T_i \text{, assigned to machine } M_k \\ & \text{is subject to starting condition j} \\ -\infty & \text{otherwise} \end{cases}$$

Now, we are in the position to write our formal state equation, in the usual minimax algebra formalism, which essentially represents the set of constraints imposed by our nominal solution on the earliest completion times of the tasks under consideration (let $\underline{x}$ be the R-dimensional vector collecting these quantities). Due to the fact that our model and the nominal solution can be relevant to a certain time instant t (in a real-time environment), from now on all considered vectors and matrices will be affected by index t. Then, the state equation can be simply written as (see [1, 3])

$$\underline{x}_t = F_t \cdot \underline{x}_t + B_t \cdot \underline{s}_t + G_t \cdot \underline{m}_t \qquad (2.1)$$

3. REPRESENTING ALTERNATIVE SOLUTIONS

In this section we want to introduce some degrees of freedom in respect of the nominal solution considered in the previous section, especially referring to assignment and sequencing of tasks.

In doing so, our aim is to represent in a compact form a number of feasible solutions which are 'perturbed' with respect to the nominal one. More specifically, we want to represent such solutions altogether by means of only one equation, where some dependences on some decisional parameters are required.

To achieve such a representation, first of all we have to specify what data structures we need so as to represent as easily as possible the nominal solution and the possible 'perturbed' ones. The data structures relevant to the nominal solution have been completely specified in the previous section, since the whole of information is provided by the matrices A, F and G.

As far as it regards the possible perturbations to the nominal solution, we have to distinguish first the tasks that can be moved (and so 'perturbed') from the ones that cannot. For the sake of simplicity, let us assume that the tasks which can be moved can be affected by only two different classes of perturbations. More complex situations can be handled with a slight complication of the formalism introduced in this section. More specifically, a task can undergo an 'assignment' perturbation, which consists in its being moved from the machine which it has been assigned to in the nominal solution to another machine, or a 'sequencing' perturbation, which models its movement from its nominal position to another one on the same machine. Of course, it is impossible that both kinds of perturbations affect a single task at once, nevertheless it is worth representing the case in which both of them are feasible. We model the possible assignment perturbation of task T_i by means of a boolean variable y_i (y_i=1 if the perturbation is accomplished), whereas w_i represents the possible sequencing perturbation of the same task.

So as to describe without any ambiguity the perturbations which a task can undergo, and the relevant alternative positions it can take, we esplicitely look upon its position in the nominal solution as an absolute reference, which means that every position a task can take as a result of some perturbation has to be defined with respect to its nominal one. What is more, if some perturbations can make some tasks become neighbours on some machine M_k, their possible sequencing on such a machine must be a-priori taken into account.

Up to this point, we have specified what information is needed in order to fully describe the possible perturbations. Now, the point is how to 'build' the matrix F so as to make its elements take into account all the allowed perturbations. We shall do this by picking up each task and analizing its precedence relationships with respect to every other task in both the 'nominal' and the 'perturbed' situations. Then, considering two tasks T_i and T_j, and with reference to the relevant element f_{ij} of F, we have to distinguish three cases a), b) and c), according to the values of the relevant elements of the matrix A, namely a_{ij} and a_{ji}.

a) $a_{ij}=a_{ji}=0$

These values mean that task T_i is not an immediate successor of T_j in the processing sequence of any job, and viceversa. In this case, we have to find out if there exists any possible configuration where T_i succeeds T_j on some machine. First of all, starting from their nominal positions, we check if there exists any 'perturbed' solution which makes the two tasks under consideration be assigned to the same machine. Due to the kinds of perturbations we allow, such machines can be at most two in number. Calling them M_k and M_h, and considering the possible mutual positions of tasks T_i and T_j we have,

$$f_{ij} = \begin{cases} t_{ik} & \text{if any configuration which makes } T_i \\ & \text{be an immediate successor of } T_j \text{ on } M_k \text{ occurs} \\ t_{ih} & \text{if any configuration which makes } T_i \\ & \text{be an immediate successor of } T_j \text{ on } M_h \text{ occurs} \\ -\infty & \text{otherwise} \end{cases} \tag{3.1}$$

We must remark now that the conditions appearing in (3.1) can be expressed as boolean functions of the vectors $\underline{y}$ and $\underline{w}$, which collect the variables representing the admissible perturbations.

So, with the agreement that $-\infty \cdot 0 = 0$, we can rewrite (3.1), using standard algebra, as

$$f_{ij} = t_{ik} \cdot \gamma_k(\underline{y},\underline{w}) + t_{ih} \cdot \gamma_h(\underline{y},\underline{w}) - \infty \cdot \overline{\gamma}_k(\underline{y},\underline{w}) \cdot \overline{\gamma}_h(\underline{y},\underline{w}) \tag{3.2}$$

where

$$\gamma_k(\underline{y},\underline{w}) = \begin{cases} 1 & \text{for any realization of the binary variables which makes } T_i \\ & \text{be an immediate successor of } T_j \text{ on } M_k \\ 0 & \text{otherwise} \end{cases}$$

and

$$\gamma_h(\underline{y},\underline{w}) = \begin{cases} 1 & \text{for any realization of the binary variables which makes } T_i \\ & \text{be an immediate successor of } T_j \text{ on } M_h \\ 0 & \text{otherwise} \end{cases}$$

Obviously, $\gamma_k(\underline{y},\underline{w})\cdot\gamma_h(\underline{y},\underline{w}) = 0$

b) $a_{ji}=1$

This means that T_j succeeds T_i in the processing sequence of some job, so $f_{ij}=-\infty$.

c) $a_{ij}=1$

This means that T_i succeeds T_j in the processing sequence of some job, so, using again standard algebra,

$$f_{ij} = t_{iz}\cdot y_i + t_{is}\cdot\overline{y}_i$$

where, assuming that T_i is a task subject to an assignment perturbation associated with the binary variable y_i, M_z is the machine to which T_i is assigned in the 'nominal' solution, whereas M_s is the alternative machine.

Of course, the pairs of elements of A which are simmetric with respect to its diagonal cannot be both equal to 1, so the case $a_{ij}=a_{ji}=1$ is infeasible.

Finally, we have to specify how to 'build' the elements of the matrix G. Such elements can be simply obtained from the data structures and each of them is dependent on only one decisional variable. In fact, for all tasks which cannot be affected by assignment perturbations, the element g_{ik} is just the same as in the previous section. On the other hand, if T_i can undergo an assignment perturbation (represented by the decision variable y_i), we have

$$g_{ik} = t_{ik}\cdot y_i - \infty\cdot\overline{y}_i$$

and

$$g_{is} = t_{is}\cdot\overline{y}_i - \infty\cdot y_i$$

where we assume that T_i is assigned to M_k in the nominal solution and to M_s in the perturbed one.

4. PERFORMANCE ANALYSIS AND OPTIMIZATION PROBLEMS

We intend now to show how to use the state equation in order to make the dependences of some classes of possible performance indexes on the considered binary decision variables explicit.

Then, we refer to equation (2.1), which is

$$\underline{x}_t = F_t(\underline{y},\underline{w}) \cdot \underline{x}_t + B_t \cdot \underline{s}_t + G_t(\underline{y},\underline{w}) \cdot \underline{m}_t$$

where now we know the matrices F_t and G_t are dependent on the decision variables. We have shown in the previous section that the dependences the matrices F_t and G_t bear on the decision variables y_k and w_j consist in some of their elements being pseudo-boolean functions of the binary decision variables.

It can be proven that the solution to the state equation (2.1) is given by

$$\underline{x}_t = R_t \cdot \{ B_t \cdot \underline{s}_t + G_t \cdot \underline{m}_t \} \tag{4.1}$$

being R_t the (RxR) resolvent matrix such that

$$R_t = E + F_t + F_t^2 .... + F_t^n \tag{4.2}$$

where n is the maximum topological distance between two tasks in the augmented precedence-relation graph. This graph is defined as the original precedence-relation one, plus the links deriving from precedence relations which can result, in any of the perturbed solutions, from the assignment of the two tasks to the same machine in immediate succession. E is the 'identity' matrix in the minimax algebra sense, whose generic element is defined as

$$e_{ij} = \begin{cases} 0 & \text{if } i=j \\ -\infty & \text{otherwise} \end{cases}$$

Let us comment on the practical usage of the result corresponding to (4.1). First of all, it is worth mentioning that, in order to obtain matrix (4.2) and then apply equation (4.1), it is necessary to make use of some symbolic computational technique so as to cope with parametric matrices.

Then, the first practical application that can be devised for (4.1) is the 'global predictability' of the effects of the simultaneous application of one or more considered perturbations on the quantities gathered in vector $\underline{x}_t$. That is, in a quite compact form, all decisional options are simultaneously taken into account and every need of detailed discrete-event simulation is completely ruled out. Actually, the computation of (4.2) and the use of (4.1), in connection with several possible choices of the set of perturbations to apply, can prove too time-consuming to let us think of an application of such 'predictability property' in a real-time environment. Nevertheless, what pointed out above is still of conceptual and practical interest.

A second possible application is relevant to the possibility of writing some overall performance index as a function of both the earliest completion times and the binary decision variables associated with the feasible perturbations.

Clearly, not every possible performance index can be written as function of $\underline{x}_t$ only. Makespan, maximum and total tardiness can, whereas, for instance, maximum and total earliness cannot, since they are not regular performance indexes. In fact, if makespan or tardiness-related costs are considered, there is no reason for not putting tasks into execution as soon as possible, compatibly with the established sequencing and assignment choices. Thus, the earliest completion times include all the information needed to compute the considered costs. On the other hand, if we chose any earliness-related cost, information relevant to the latest completion times should be given, by means of a state equation similar to that considered in this paper. The investigation of such problems will be matter of future work.

Let us now come back to costs related to the makespan (maximum task completion time) or to the tardiness ($\max\{x_t(i) - d(i), 0\}$, where $x_t(i)$ is the earliest and actual completion time of task T_i, and d(i) is its due-date). Then, to give an example, we can write the makespan as

$$M(\underline{y},\underline{w}) = \max_{i=1,..,R} x_t(i) \tag{4.3}$$

Then, each $x_t(i)$ can be computed by means of (4.1), and, taking into account the meaning of the matrix product in the minimax algebra environment, we can write, in the usual algebra notation,

$$x_t(i) = \max \{f_1(\underline{y},\underline{w}), ..., f_R(\underline{y},\underline{w})\} \tag{4.4}$$

where each term $f_i(\underline{y},\underline{w})$, $i = 1,..., R$, is obtained as the (ordinary) sum of an element of matrix R_t and an element of vector $\{ B_t \cdot \underline{s}_t + G_t \cdot \underline{m}_t \}$. This example helps to clarify the structure of the cost functional $M(\underline{y},\underline{w})$ that arises in this case. Similar considerations can be made for tardiness-related costs. The binary (unconstrained) optimization problems that have to be faced have a general structure similar to that of the cost functional that can be deduced from (4.2), (4.3) and (4.4).

5. CONCLUSIONS

We have presented a general approach to analyze performances and state optimization problems in discrete-event systems characterized by a fairly general structure where specific choices about the assignment and the sequencing of tasks are to be made.

First, we have shown how to determine such a state equation of the discrete-event system to represent every possible decisional option. Then, we have seen that it is possible to obtain a direct dependence of certain performance indexes on the decisional variables under concern by using the minimax algebra resolvent matrix. This possibility makes our approach in some sense related to the perturbation analysis technique [4]. Moreover, it is possible to state, in terms of the considered variables, optimization problems having the type of structure specified in the last section.

At this point, it is worth doing some remarks. First, the determination of the state equation taking into account the whole set of the possible alternatives can be a time-consuming, although conceptually simple, task. In turn, as already pointed out, symbolic computation techniques are required in order to compute the resolvent matrix, as it is dependent on the binary decision variables. Then, once the expression of the optimization objective as a function of the decision variables has been written, inspection techniques, essentially based on the recognition of those binary variables which have no effect on the considered objective, can be devised to simplify such an expression. Finally, efficient techniques for the resolution of the resulting combinatorial optimization problems are still under study.

REFERENCES

[1] Cuninghame-Green, R., *Minimax Algebra.* Lecture Notes in Economics and Mathematical Systems, Vol.166, Springer-Verlag, Berlin, 1979.

[2] Cohen, G., D. Dubois, J.P. Quadrat, and M. Viot, *A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing.* IEEE Trans. Automat. Control, Vol. AC-30, 1985, pp.210-220.

[3] Engell, S., *Predictive control of Flexible Manufacturing Systems.* Proc. 28th Conf. on Decision and Control, Tampa, FL, 1989, pp.54-59.

[4] Ho, Y.C., *Performance evaluation and perturbation analysis of Discrete Event Dynamic Systems.* IEEE Trans. Automat. Control, Vol. AC-32, 1987, pp.563-572.

[5] French, S., *Sequencing and Scheduling: an Introduction to the Mathematics of the Job Shop*, Wiley, New York, 1982.

Dynamics of a Switched N Buffer System

Christopher Chase, Peter Ramadge *
Department of Electrical Engineering
Princeton University, Princeton, N.J., 08544

Abstract

We analyze the dynamical behavior of a flexible manufacturing system model originally posed by Perkins and Kumar [1]. This model has N buffers with a single server that is switched from buffer to buffer by a local controller. This is an example of a hybrid continuous variable and discrete event system. A metric is defined for feedback controllers. The main result is that the feedback sequence is eventually periodic and the buffer states are asymptotically periodic for an open dense set in the space of feedback controllers.

1 Introduction

This work was motivated by a model for flexible manufacturing system originally proposed by Perkins and Kumar [1]. However, the authors feel that the results presented here for the Perkins and Kumar model extend to more a general class of discrete time systems that use a discrete feedback controller to select from a set of transition functions. In [2] we have presented results analogous to those presented here for systems that implement a finite number of one-dimensional transition maps via a simple feedback controller. The system and results considered herein should be taken as simple examples of typical behavior to be exhibited by more general N-dimensional systems.

The system model consists of N buffers, with work arriving to buffer i at a constant rate of $\rho_i > 0$, and a switchable server that removes work from any selected buffer at unit rate. When the server is removing work from buffer j, we say that the server is in location j. The location of the server is a control variable and can be chosen using a feedback policy. For this paper we consider the simple case when the system is closed, so that $\sum_i \rho_i = 1$, and switching takes place instantaneously [see Figure 1 for $N = 3$].

We consider a class of feedback or supervision policies called clearing policies that have the following properties. The server remains in its current location until the associated buffer empties. Then the server instantaneously switches to a new buffer given by a deterministic function of the current state.

The continuous time system can be modeled completely by a discrete time system. In this case there are linear maps $F_i : X \to X$ that describe the clearing of buffer i. Let $S : X \to \{1, \ldots, N\}$ be the feedback map that specifies the next buffer to serve at each clearing time. Then the state transition function of the sampled system is the function $F : X \to X$ with

$$F(x) = F_{S(x)}(x)$$

This particular system is representative of systems in which multiple tasks may be performed, represented by the different update maps, and these tasks must be scheduled as the system evolves. The dynamics of the system exhibits multiple modes (the different update maps) which must be scheduled by a controller. From this perspective, this work continues that initiated in [3].

---

[*] Research partially supported by NASA under contract NAG 2-558

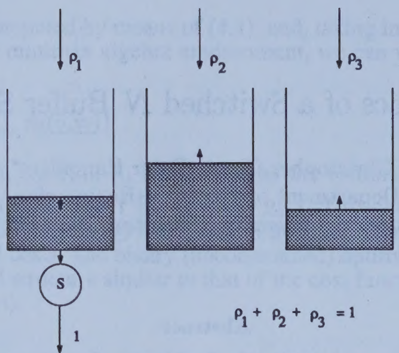


Figure 1: The switched server system.

There have been many studies of scheduling actions. For example, the multi-armed bandit problem [4, chapter nine and references] and the feedback control of flexible manufacturing systems [1]. Often one is interested in achieving an optimum schedule in terms of some performance criterion, and many times, such as in the multi-armed bandit problems, the schedule is a function of the current state, i.e., it is determined by a feedback policy.

In this paper we do not explore optimum scheduling but rather we investigate the complex dynamics that can result from scheduling or supervising multi-mode systems. The type of scheduling we investigate here can be viewed as a quantized feedback of the current state. Such a system is a hybrid continuous variable and discrete variable system. Recent work by Delchamps [5] has investigated the induced dynamics of a class of such hybrid systems. Overstepping traditional control techniques, he has demonstrated that such systems can exhibit chaotic but statistically stable behavior. In addition, there are a number of studies on expansive maps of the unit interval [6,7] which demonstrate chaotic behavior. Of particular interest is [8] wherein the dual system of the one considered here is examined. These systems are amenable to statistical analysis. In contrast, the switched server system investigated here is piecewise nonexpansive. Our purpose here is to show that this system can be viewed as having "nice" predictable behavior.

The feedback controller is a function of the current state which we also call a switching function. After giving a complete definition of the system, we define a metric on the space of all possible switching functions. Next we show that the system dynamics can be modeled at a higher level via the observability congruence. Conditions are given when this congruence yields a finite state automaton model for the system. To meet these conditions we first show that the switching functions of interest generate control sequences that make the system piecewise contractive. This is used to show the "generic" predictability of these systems.

2 System Definition

We refer to the multiple buffer server model as the switched server system (SSS). Let $[1, N]$ stand for the set $\{1, \ldots, N\}$. Take N to be the number of buffers with work arriving in buffer i at rate ρ_i, for $i \in [1, N]$. Set the system parameter $\rho = [\rho_1, \ldots, \rho_N]$. Let $w_i(t)$ be the work in buffer i and let the system state be $w(t) = [w_1(t), \ldots, w_N(t)]^T$. For the closed SSS that we consider here, $\|\rho\|_1 = 1$. Thus the total work in the buffers remains constant. We will assume that $\|w(t)\|_1 = 1$. As mentioned earlier, we adopt a control strategy or policy that the server clears a buffer at a given location then switches to a new location. Such control policies are called clearing policies.

Note that while the server remains at a nonempty buffer the state, $w(t)$, changes linearly with time. Thus the state trajectory is completely determined by its values at the switching times. Let $\{t_i\}_0^\infty$ be the sequence of switching times with $t_0 = 0$. We define a discrete time model by sampling the continuous time system at switching times which for clearing policies are the times at which a buffer empties. For the discrete time system the state at time n is $x(n) = [x_1(n), \dots, x_p(n)]^T = w(t_n)$. The state takes values in the state space

$$X = \{x \in \mathrm{IR}^N | \ \|x\|_1 = 1, \ x_i \geq 0, \ i = 1, \dots, N\}$$

We identify those regions of the state space where a particular coordinate is zero by the sets Δ_i, $i \in [1, N]$, where $\Delta_i = \{x \in X | \ x_i = 0\}$. In general, X is a multi-dimensional tetrahedron and the Δ_i are its faces.

Suppose at the n^{th} switch the state is x and the server moves to location i. Let $F_i : X \to X$, $i = 1, \dots, N$, be the map describing the clearing of buffer i. Then $F_i(x) = A_i x$, where A_i is a $N \times N$ matrix of the form:

$$A_i = \begin{bmatrix} 1 & & & \frac{\rho_1}{1-\rho_i} & & & \\ & \ddots & & \vdots & & & \\ & & 1 & \frac{\rho_{i-1}}{1-\rho_i} & & & \\ & & & 0 & & & \\ & & & \frac{\rho_{i+1}}{1-\rho_i} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & \frac{\rho_P}{1-\rho_i} & & & 1 \end{bmatrix}$$

Note that F_i is a projection onto Δ_i. Let $\{c_n\}_1^N$ be the columns of A_i. Then

$$\|c_i\|_1 = \frac{\sum_{n \neq i} \rho_n}{1 - \rho_i} = \frac{\sum_n \rho_n - \rho_i}{1 - \rho_i} = 1.$$

The induced 1-norm of A_1 is $\|A_1\|_1 = \max_i \|c_i\|_1 = 1$. So F_i is nonexpansive in the 1-norm.

Although the nullspace of A_i is one-dimensional, we can compute an inverse for F_i since it is restricted to the set X. We assume that F_i is not applied to points in Δ_i, because once a buffer is cleared a different buffer must be cleared next. Let x' be a preimage of x, i.e. $F_i(x') = x$. We can compute

$$x'_j = \ x_j - \tfrac{\rho_j}{1-\rho_i} x'_i \quad \text{for } j \neq i \tag{1}$$

with x'_i the free parameter. Since x' must have at least one coordinate equal to zero other than x'_i it can only take at most $N - 1$ values. Furthermore, equation (1) is decreasing in x'_i, so x'_i must be taken as the minimum of the possible values to ensure that all coordinates are non-negative. Then we find that the unique inverse of F_i is a map $F_i^{-1} : \ \Delta_i \to \cup_{j \neq i} \Delta_j$ given by

$$F(x)_j = \begin{cases} x_j - \frac{\rho_j}{\rho_n} x_n & j \neq i \\ \frac{1-\rho_i}{\rho_n} x_n & j = i \end{cases} \tag{2}$$

where $n \neq i$ is any argument minimizing x_j / ρ_j.

In this paper we investigate feedback controllers that are a function of the current state. For this problem, a current state feedback function, called a switching function, is a map $S : \ X \to \{1, \dots, N\}$. Given a switching function S, the update map for the discrete time system is given by a function $F : \ X \to X$ with $F(x) = F_{s(x)}(x)$. Then for an initial condition $x(0) \in X$ the state trajectory is given by $x(n + 1) = F(x(n))$

To help understand the long term system dynamics, it is convenient to define the forward attractor, Λ. This is the set of limit points of all possible trajectories. Fix a switching function S, and consider the images of X under iterations of F. Then

$$\Lambda = \cap_{n=1}^\infty \overline{F^n(X)}$$

Note that $F^{n+1}(X) \subset F^n(X)$ for all n.

3 Switching Functions

The focus of this paper is to analyze the typical behavior when multi-modal systems use current state feedback controllers which we call switching functions. Let $\mathcal{S}$ denote the set or space of all switching functions. For an arbitrary switching function $S\colon X \to [1, N]$ the sets $S_i = S^{-1}(i)$ for $i \in [1, N]$ are called the switching sets. A switching function is completely defined by its switching sets. Difficulties in analyzing the SSS server arise from discontinuities introduced in the closed-loop system F at the boundaries of the switching sets. These boundary points will play an important role in our analysis. So we define the set ∂S of switching points or switching boundary as the complement of points interior to the switching sets, i.e.

$$\partial S = X \setminus \cup_{i=1}^{N} \operatorname{int}(S_i)$$

Note ∂S is a compact set. For practical purposes the switching boundary will be a very simple set.

3.1 Metric for Controllers

We define a metric on a certain partition of $\mathcal{S}$. We use the Hausdorff distance to naturally introduce a metric on switching functions. For $x \in X$, and nonempty $B \subset X$ define $d(x, B) = \inf\{|x - y| : y \in B\}$. Then $d(x, A)$ is the distance from the point x to the set A. For nonempty $A, B \subset X$, define $d(A, B) = \sup\{d(x, B) : x \in A\}$. So, $d(A, B)$ is the distance from the set A to the set B. The Hausdorff distance between two nonempty sets $A, B \subset X$ is $h(A, B) = \max\{d(A, B), d(B, A)\}$. If either set is empty, set $h(A, B) = \infty$ and if both are empty, set $h(A, B) = \infty$.

Let $\mathcal{H}(X)$ be the collection of nonempty compact sets in X. Then it is not difficult to show that h is a metric on $\mathcal{H}(X)$ and $\mathcal{H}(X)$ is complete with respect to h (see for example [9]). Note that since X is bounded and $h(A, B) = h(\overline{A}, \overline{B})$, we can view h as a metric on a partition of 2^X. This is the natural partition induced on the subsets of X by the set closure operation. Let $\mathcal{H}(X)^N$ be the space of N-tuples over $\mathcal{H}(X)$. For $A = (A_1, \ldots, A_N)$ and $B = (B_1, \ldots, B_N)$ in $\mathcal{H}(X)^N$, define

$$h_1(A, B) = h(A_1, B_1) + \ldots + h(A_N, B_N)$$

Clearly h_1 is a metric and $\mathcal{H}(X)^N$ is complete with respect to it. Again via the set closure operation, we can view h_1 as a metric on a partition of $(2^X)^N$, the space of N-tuples over subsets of X. If we identify a switching function with the N-tuple formed by its switching sets, i.e. $(S_1, \ldots, S_N)$, then the space of all switching functions, $\mathcal{S}$, is embedded in $(2^X)^N$. In fact, every $(A_1, \ldots, A_N)$ in $(2^X)^N$ where $\{A_i\}$ forms a partition of X is a switching function. In addition, h_1 is a metric on a partition of $\mathcal{S}$ where two switching functions S and V are equivalent if $\overline{S_i} = \overline{V_i}$ for $i \in [1, N]$. Let c stand for the equivalence relation on $(2^X)^N$ induced by the closure operation applied component-wise to elments of $(2^X)^N$. Then $\mathcal{S}/c$ denotes the quotient set of switching functions in which we are interested.

We would like to make some comments on using the metric space $(\mathcal{S}/c, h_1)$. If two switching functions are equivalent then they have the same boundary points and may differ only on those boundary points. Thus the dynamics of these systems will only differ for initial conditions in the set of boundary points. Note that it is possible that each switching set is dense in X in which case $\partial S = X$ and it can be shown that $\Lambda(S) = X$. However, for implementation purposes one would expect the switching sets to have nonempty interior to allow practical calculations of the switching function. In addition, one would expect the switching regions to not contain isolated points and to have a finite number of connected components. Indeed, an easily implemented switching function would use very simple boundary regions to actually determine membership in a particular region. For example, the Clear-Largest-Work switching function has hyperlanes for the boundary regions allowing fast and easy computation.

Another comment about the metric h_1: given any switching function S, we call any switching function that has distance less than $\delta \geq 0$ from S to be a δ-perturbation of S. Perturbations with respect to h_1 make more sense when the switching regions have nonempty interiors and simple boundaries. Although we will not make any more restrictions on switching functions outside those given in the next section the comments regarding practical switching functions should be kept in mind.

3.2 Regular Controllers

In [1] a notion of stability is defined for the SSS model. This notion guarantees a realistic performance for the system. For the closed system the appropriate notion of stability is the bounded frequency of switches. This requires that for a given switching function and any initial condition the number of complete switching cycles in a fixed length of time remain bounded. Therefore, define $N(t, T, x_0, S)$ to be the number of complete switches (i.e. the number of switches after a possible first switch) in the open interval $(t, t+T)$ for initial condition x_0 and switching function S. Now we say that a switching function for the system is stable if the local switching frequency is finite, that is, if

$$\sup_{t \geq 0, T > 0, x_0} \frac{N(t, T, x_0, \sigma)}{T} < \infty. \tag{3}$$

Another equivalent definition is that the time between switches remain bounded away from zero. This interpretation of stability guarantees that a minimum amount of work is done every time a buffer is served.

Naturally we are interested in implementing stable switching functions. To help characterize stable switching functions we define two classes of switching functions. A switching function is *Clear-A-Fraction (CAF)* if it guarantees a minimum amount of work is done every time a buffer is cleared. Precisely, there exists some $\alpha > 0$ such that for all $x \in X$, $x_{S(x)} \geq \alpha$.

We also define a *complete* switching function. First, a finite sequence in $[1, N]$ is complete if it contains every number in $[1, N]$. An infinite sequence $\{p_n\}_0^\infty$ in $[1, N]$ is complete if there exists an integer K, such that $\{p_i, p_{i+1}, \cdots, p_{i+K-1}\}$ is complete for every $i \geq 0$. For any such K, the sequence $\{p_n\}$ is also called K-complete. A switching function S is complete (K-complete) for a given system if for any initial condition the observed buffer sequence is complete (K-complete).

There is a tight relationship between the stable, CAF, and complete properties of switching functions. First, a switching function S is stable if and only if S is Clear-A-Fraction (CAF). This is immediate in light of the definition of stability and since the total work in the system is constant. As a result, stability is independent of the system parameter because the CAF property depends only on the switching function. On the other hand, there is a near equivalence between the CAF and K-completeness properties. The following is easily proved [10]:

Lemma 1 Every CAF switching function is K-complete. Every K-complete switching function will be CAF eventually meaning that there is a $t_0 \geq 0$ such that equation 3 holds for $t \geq t_0$.

In fact, if a switching function is CAF with fraction α then it is $K(\alpha)$-complete where $K(\alpha) = \left\lceil \frac{1}{\rho_{\min}\alpha} \right\rceil$. We note that if some switching function S is CAF with fraction α then any δ-perturbation of S for $\delta < \alpha/2$ is $K(\alpha)$-complete.

4 Observability Congruence

For a feedback map S, the switching sets partition the state space X into N equivalence classes. Let this equivalence relation on X be denoted by π_0. Define a sequence of equivalence relations π_n by

$$x \equiv y(\bmod \pi_n) \iff F(x) \equiv F(y)(\bmod \pi_{n-1})$$

The lattice of equivalence relations on a fixed set X is complete. Thus the relation $\pi = \wedge_{n=0}^{\infty}\pi_n$ is an equivalence relation on X. It is the unique coarsest equivalence relation on X that is finer than every π_n.

It is usual to refer to π as the *observability congruence* of the system (X, F, S), since $x \equiv y(\bmod \pi)$ iff for all $n \geq 1$, $S(F^n(x)) = S(F^n(y))$, i.e., x and y produce the same feedback or observation sequence. Moreover, it is easy to see that π is a congruence on (X, F), i.e., that $x \equiv y(\bmod \pi)$ implies that $F(x) \equiv F(y)(\bmod \pi)$.

Let $\hat{X}$ denote the set of equivalences classes of π, and let π also denote the canonical projection $\pi : X \to \hat{X}$. Then there exists a unique map $\hat{F} : \hat{X} \to \hat{X}$ such that $\pi F = \hat{F}\pi$. Since π is finer than π_0, and $x \equiv y(\bmod \pi_0)$ iff $S(x) = S(y)$, it is clear that there exists a map $\hat{S} : \hat{X} \to \{1, \ldots, N\}$ such that $S = \hat{S}\pi$. The above observations are summarized in the commutative diagram:

$$\begin{tikzcd}
X \arrow[r, "F"] \arrow[d, "\pi"'] & X \arrow[r, "S"] \arrow[d, "\pi"] & \{1, \ldots, N\} \\
\hat{X} \arrow[r, "\hat{F}"] & \hat{X} \arrow[ur, "\hat{S}"'] &
\end{tikzcd}$$

The original system (X, F, S) is said to be *reduced* to the quotient system $(\hat{X}, \hat{F}, \hat{S})$, or to be the *preimage* of the quotient system.

The most interesting situation is when the quotient system is considerably simpler in structure than the original system. For example, if the quotient system has a finite number of states, then the continuous variable nonlinear system will be reduced to a finite automaton. In this case the switching sequence, and hence the supervision policy, will be completely determined by a deterministic finite state system. As a result every feedback sequence will be eventually periodic. Notice, moreover, that the automaton will display both the transient as well as the steady state behavior of the switching sequence.

It is precisely the situation when the quotient system is a finite automaton in which we are interested. This is equivalent to saying that π has only a finite number of cosets, i.e., that π has finite index.

Theorem 1 If Λ does not contain any switching points then the observability congruence π has finite index.

Proof: Note that since X is compact and the boundary set is closed it is easy to see that if Λ does not intersect the boundary set, ∂S, then there exists a positive integer M such that $\overline{F^M(X)}$ does not intersect ∂S. Let μ be the minimum distance between $\overline{F^M(X)}$ and the boundary set, i.e.

$$\mu = \inf\left\{ \|x - y\|_1 \mid x \in \overline{F^M(X)}, y \in \partial S \right\}$$

This distance is nonzero because the sets involved are compact. Let $B_\mu(x)$ be the open ball of radius μ (with respect to the $\| \ \|_1$) centered at x. Then there exists a finite collection of such open balls $C = \{B_\mu(x^i)\}_{i=1}^{J}$ that covers $\overline{F^M(X)}$ with each ball centered at some point in $\overline{F^M(X)}$. Also, F is continuous on $\overline{F^M(X)}$ since $\overline{F^M(X)}$ is bounded away from the boundary set. Therefore,

$$F(\overline{F^M(X)}) \subset \overline{F(F^M(X))} \subset \overline{F^M(X)}$$

i.e. $\overline{F^M(X)}$ is invariant under F.

Let B be a ball from C. B is interior to a single switching set since it intersects $\overline{F^M(X)}$ and has radius μ. Recall that the F_i are nonexpansive in the $\| \ \|_1$ norm. Thus $\overline{F(B)} = F_j(B)$ for some j implying that $F(B)$ has radius less than or equal to μ and intersects $\overline{F^M(X)}$. Similarly,

$F^n(B)$, for $n = 2, 3, \ldots$, has radius less than or equal to μ and intersects $\overline{F^M(X)}$. This implies that for $x, y \in X$ with $F^M(x), F^M(y) \in B$ then

$$S(F^n(x)) = S(F^n(y)) \quad \text{for } n \geq M$$

As there are only J sets in the cover C, there can only be J distinct buffer sequences after M switches. This implies that S can produce only a finite number of distinct buffer sequences. Since each equivalence class in π corresponds to a unique buffer sequence, π must have finite index. ∎

5 Properties of K-complete Sequences

We will show that stable switching functions possess a local contraction property using the K-completeness property. Fix K and for each K-complete sequence $\{s_i\}_1^K$ define the map:

$$H_k = \prod_{i=1}^{N} F_{s_i} \tag{4}$$

with each $k = 1, 2, \ldots$ indicating a different sequence (of which there are a finite number). Furthermore, let $L_k = s_K$ of the corresponding sequence. We will not allow sequences with repeated consecutive numbers, e.g. 3 1 1 2 or 2 2 3 1, since a buffer can not be cleared twice in a row for a stable switching function.

Let V_i be the set of vertex points of Δ_i, $V_i = \{v \in \Delta_i | \; v_k = 1 \text{ for some } k \neq i\}$. Let $\partial\Delta_i$ be the boundary of Δ_i and let $\overset{\circ}{\Delta}_i$ be the interior of Δ_i.

Lemma 2 $H_k(V_i) \subset \overset{\circ}{\Delta}_{L_k}$ for all i and k.

Proof: Note that if $x_i \neq 0$ then $A_i x \in \overset{\circ}{\Delta}_i$. Let $v \in V_i$. Let s_j be the first number in the clearing sequence for H_k with $x_{s_j} \neq 0$. Then $\prod_{i=1}^{j-1} A_{s_i} x = x$ and $A_{s_j} x \in \overset{\circ}{\Delta}_{s_j}$. Therefore,

$$\prod_{i=1}^{l} A_{s_i} x \in \overset{\circ}{\Delta}_{s_l} \quad \text{for each } l, i \leq l \leq K$$

∎

Since $H_k(V_i)$ contains only a finite number of points and is interior to Δ_{L_k}, Δ_{L_k} can shrink a bit and still contain $H_k(V_i)$. More precisely, define for $i \in [1, N]$ and each k

$$\gamma_{ki} = \min\{\gamma : H_k(V_i) \in \gamma\Delta_{L_k} + b, \text{ for some } b\}$$

The minimum exists and $\gamma_{ki} < 1$. Let $\gamma = \max_{ki} \gamma_{ki}$. And since the maximum is over a finite set, $\gamma < 1$. Note, however, that γ will in general depend on K (in fact, will be nondecreasing with K.) Let $co(B)$ denote the convex hull of B. As $\gamma\Delta_{L_k} + b$ is convex and H_k is linear, it follows that

$$H_k(\Delta_i) = co(H_k(V_i)) \subset \gamma\Delta_{L_k} + b_{ki}$$

for an appropriate b_{ki}. By linearity we have

$$H_{k_2} H_{k_1}(\Delta_i) \subset \gamma^2 \Delta_{L_{k_2}} + c$$

for some $c \in \mathbb{R}^N$. By repeated application of this for some sequence $\{H_{k_j}\}_1^\infty$,

$$\prod_{j=1}^{n} H_{k_j}(\Delta_i) \subset \gamma^n \Delta_{L_{k_n}} + c \quad \text{for some } c \in \mathbb{R}^N \tag{5}$$

Taking the diameter of the sets in the above expression we have

$$\lim_{n\to\infty} \operatorname{diam} \prod_{j=1}^{n} H_{k_j}(\Delta_i) \leq \lim_{n\to\infty} \gamma^n \operatorname{diam}\Delta_{L_{k_n}} = 0 \tag{6}$$

Now we have the following contraction property of K-complete sequences.

Theorem 2 Given the parameter ρ, $\|\rho\|_1 = 1$ and a map H corresponding to an K-complete sequence as defined in (4), H^n is a contraction in the 1-norm on each Δ_i for some n that depends only on γ.

Proof: The metric we consider here is the 1-norm, but the proof holds for any compatible metric. Consider an open ball B_r, of radius r in Δ_i. This ball is contained in some translated, scaled version of Δ_i, call it E, with diameter cr for some constant c. That is, $E = a\Delta_i + b$ with $a = cr/\operatorname{diam}\delta_i$ and some b. Take n such that $c\gamma^n < 1$. Then by the linearity of H and using equation (6),

$$\operatorname{diam}H^n(B_r) \leq \operatorname{diam}H^n(E) \leq a\gamma^n \operatorname{diam}\Delta_{s_K} < r = \operatorname{diam}B_r$$

which is sufficient to show that H^n is a contraction. ∎

We desire to make one more comment regarding buffer sequences. Note that the interiors of $F_j(\Delta_i)$ and $F_k(\Delta_i)$ are disjoint when $j \neq k$. Furthermore, because F_i is invertible for each i the image under F_i of two disjoint sets in $X \setminus \Delta_i$ are disjoint. Then if $\{s_i\}_1^M$ and $\{\bar{s}_i\}_1^M$ are two distinct buffer sequences,

$$\operatorname{int}\left(\prod_{i=1}^{M} F_{s_i}(\Delta_j)\right) \cap \operatorname{int}\left(\prod_{i=1}^{M} F_{\bar{s}_i}(\Delta_j)\right) = \emptyset \tag{7}$$

for j not equal to s_1 or $\bar{s}_1$. This small fact will be used in proving our main result in the next section.

6 Generic Predictability

We now present our main result showing that the SSS with a stable switching function generically exhibits predictable behavior. First we present an easily proved lemma regarding switching function perturbations within a limited region of the state space. For a set B in X and S in $(2^X)^N$, let $S|B$ be the restriction of S to B defined by

$$S|B = (S_1 \cap B, \ldots, S_N \cap B)$$

Lemma 3 [10] Let $\{B_i\}_1^M$ be a collection of disjoint subsets in X. Let S and S' be switching functions such that their symmetric difference is restricted to the B_i, i.e.

$$S \setminus S' \cup S' \setminus S \subset \cup_i B_i$$

and $h_1(S|B_i, S'|B_i) \leq \delta$ for $i \in [1, N]$. Then $h_1(S, S') \leq \delta$.

Theorem 3 For the switched server system with parameter ρ, there is an open dense set, D in the space of stable switching functions such that for each switching function in D the following are true:

1. The observability congruence π has finite index.
2. All orbits converge uniformly to periodic orbits.

Proof: All distances will be with respect to the $\| \|_1$ norm. Take any $\epsilon > 0$. Consider a stable policy S. S is CAF for some fraction $\alpha > 0$. Let δ be the minimum of $\epsilon/2$ and $\alpha/4$. S is $K(\alpha/2)$-complete. Let γ be as defined in section 5 for $K(\alpha/2)$. Take L such that $\gamma^L \mathrm{diam} X < \delta$. Let $R = F(X)$ be the range of F. For $i \in [1, N]$ define the sets

$$\begin{aligned} R_i &= R \cap \Delta_i \\ W_i &= \{x \in \Delta_i | \; x_j \geq \frac{\alpha}{2}\rho_{\min}, j \neq i\} \end{aligned}$$

Note that for any CAF switching function with fraction at least $\alpha/2$ the range is a subset of $\cup_i W_i$. So $R_i \subset W_i$.

Consider the collection $\mathcal{A}$ of $K(\alpha/2)$-complete sequences (without consecutive repetitions) of length $M = LK(\alpha/2)$. Index $\mathcal{A}$ by k. For each k, we have $\{s_i\}_1^M$ in $\mathcal{A}$ and define

$$H_k = \prod_{i=1}^{M} F_{s_i}$$

For $i \in [1, N]$ and each k let $B_{k,i} = H_k(\Delta_i)$. From the comments at the end of section 5 the interior of the B_{ki} are disjoint from each other.

We define a δ-perturbation of S, S', such that $\Lambda(S')$ contains no boundary points of S'. Let $S' = S$ outside of $\cup_{ki} \overset{\circ}{B}_{ki}$. Note that H_k is continuous and one-to-one on each Δ_i. This implies that for each k and $i \in [1, N]$, $H_k(W_i)$ is closed and interior to B_{ki}, which is also closed. Hence, for each k and $i \in [1, N]$, there exists a closed set C_{ki} interior to B_{ki} with $H_k(W_i)$ interior to C_{ki}. For each k and i let $l \in S(H_k(R_i))$ and define $S' = l$ on C_{ki}. Define S' on $\overset{\circ}{B}_{ki} \setminus C_{ki}$ such that $S'(\overset{\circ}{B}_{ki} \setminus C_{ki}) = S(\overset{\circ}{B}_{ki})$. Such a definition for S' exists since $\overset{\circ}{B}_{ki} \setminus C_{ki}$ is a nonempty open set. This completely defines S'.

By our choice of L, $\mathrm{diam} H_k(\Delta_i) < \delta$ for each k and i. Since $S'(\overset{\circ}{B}_{ki}) = S(\overset{\circ}{B}_{ki})$, we have

$$h_1(S| \, \overset{\circ}{B}_{ik}, S'| \, \overset{\circ}{B}_{ik}) \leq \delta$$

Then the previous lemma implies that $h_1(S, S') \leq \delta$. So S' is within ϵ of S. This also implies that S' is CAF for fraction $\alpha/2$. Let F' be the transition function corresponding to S'. Because S' is $K(\alpha/2)$-complete, the range of F' is inside $\cup W_i$ which is closed. Therefore, $\overline{F'^{M+1}}(X)$ is a subset of $\cup_{ki} H_{ki}(W_i)$ which is also closed. Since S' was defined such that $\cup_{ki} H_{ki}(W_i)$ is interior to the switching regions of S', $\overline{F'^{M+1}}(X)$ contains no boundary points of S'. Thus $\Lambda(S')$ contains no boundary points and the observability congruence for F' has finite index by Theorem 1.

Let μ_{ki} be the minimum distance separating $H_k(W_i)$ from ∂C_{ki}. This distance is nonzero because these two sets are compact and disjoint. Let μ equal the minimum of the μ_{ki}. Note that μ is less than $\alpha/4$. The set $E(S') = \{S \in \mathcal{S} | \; h_1(S, S') < \mu\}$ is an open ball of switching functions centered at S' that are CAF for fraction $\alpha/2$. We see that for each switching function in $E(S')$, $H_k(W_i)$ will still not contain any boundary points. Thus the observability congruence corresponding to each element of $E(S')$ is finite.

If the observability congruence is finite then all buffer sequences are periodic after some finite number of switches and the number of different periodic cycles is also finite. Any of the periodic cycles must be K-complete where K is the length of the cycle. H_k, as defined by equation 4 for a particular periodic cycle, is a contraction by Theorem 2. Under the conditions of our problem, Theorem 2.2 in [3] implies that the state trajectory for a periodic buffer sequence converges asymptotically to a periodic orbit which depends only on the actual periodic buffer cycle and not on the initial condition. This implies that the convergence is uniform, since there are only a finite number of periodic cycles and the transients of every buffer sequence are uniformly bounded.

Let $S'(\epsilon, S)$ be the switching function obtained above for ϵ and S. Define $D = \cup E(S'(\epsilon, S))$ where the union is over all $\epsilon > 0$ and all stable S. Then the set D is an open dense set in the space of switching functions that satisfies items 1 and 2. ∎

7 Conclusion

We have thoroughly investigated an example of the supervision of a continuous variable system using a discrete controller. Generically with respect to stable feedback control, the switched server system exhibits very predictable behavior which is modeled by a finite state automaton. This behavior is generic because there is at least an open dense set of such controllers in the space of stable feedback controllers.

Our work here is along the lines of that initiated in [3]. The aim of this research is to improve our ability to analyze mixed variable control systems where the controller operates on discrete events. This has application to multimodal discrete time systems where the controller can choose from a set of transition maps to implement.

Future work could try to extend these results to systems using more general nonlinear maps. These results easily extend to the SSS model with nonzero setup times as in [1]. In a different vane, it would be worthwhile to see if these type of results can be applied to networks of multiple buffer servers as modeled in this paper. Another question to be answered is if these models have any applications to traditional queueing models.

References

[1] J. R. Perkins and P. R. Kumar, "Stable, distributed, real-time, scheduling of flexible manufacturing/assembly/disassembly systems," *IEEE Trans. Automat. Contr.*, vol. 34, no. 2, pp. 139–148, February 1989.

[2] C. J. Chase and P. Ramadge, "Predictability of a class of supervised one-dimensional systems," in *Proc. 5th IEEE Inter. Symposium on Intelligent Control*, Philadelphia, Penn., September 1990.

[3] P. Ramadge, "On the periodicity of symbolic obsrvations of piecewise smooth discrete-time systems," *IEEE Trans. Automat. Contr.*, vol. 35, no. 7, pp. 807–813, July 1990.

[4] J. Walrand, *An Introduction to Queueing Networks*, Englewood Cliffs, NJ: Prentice Hall, 1988.

[5] D. F. Delchamps, "The stabilization of linear systems with quantized feedback," in *Proc. 27th IEEE Conf. Decision and Control.*, Austin, Texas, December 1988.

[6] T. Li and J. A. Yorke, "Ergodic transformations from an interval into itself," *Tran. of the Amer. Math. Society*, vol. 235, pp. 183–192, January 1978.

[7] A. Boyarsky and M. Scarowsky, "On a class of transformations which have unique absolutely continuous invariant measures," *Tran. of the Amer. Math. Society*, vol. 255, pp. 243–262, November 1979.

[8] C. J. Chase, J. Serrano, and P. Ramadge, "Periodicity and chaos from switched flow systems: contrasting examples of discretely controlled continuous systems," June 1989, in preparation for publication.

[9] M. Barnsley, *Fractals Everywhere*, Boston: Academic Press, 1988.

[10] C. J. Chase, "Multiple buffer servers," 1990, Chapter 2, Ph.D. Thesis (in preperation), Dept. of Electrical Engineering, Princeton University.

On Discrete State Realizations for Condition/Event Systems

R. S. Sreenivas and Bruce H. Krogh
Department of Electrical Engineering and Computer Engineering
Carnegie Mellon University
Schenley Park
Pittsburgh, PA 15213

ABSTRACT

Condition/event (C/E) systems are a class of continuous-time dynamic systems with two types of discrete-valued input and output signals: *condition signals* and *event signals*. In applications such as discrete control, C/E systems provide an intuitive modeling framework amenable to block diagram representation. In the context of discrete event systems theory, C/E systems provide a well-defined system paradigm with an input-output signal structure which does not require the a priori identification of input events with internal state transitions or output events. In this paper we introduce C/E systems with discrete state realizations, that is, C/E systems with internal dynamics which can be modeled in terms of a discrete state space. We develop equivalent discrete state realizations for C/E systems resulting from cascade and feedback interconnections. The paper concludes with a discussion of several directions for future research.

1 Introduction

This paper concerns a class of discrete event dynamic systems (DEDS) called *condition/event systems* (C/E systems)[1]. Figure 1 illustrates the basic input/output signal flow for a C/E system. There are two classes of input and output signals for a C/E system: *condition signals*, which are piecewise constant signals taking on values in a finite set of *conditions*; and *event signals*, which are *null* except at discrete points in time at which they assume a value in a finite set of *events*. In block diagrams, signal flow lines carrying event signals are distinguished from signal flow lines carrying condition signals by the symbol —⋀⋁►, the standard symbol for interrupt signals in real-time control-flow diagrams [2]. When the internal dynamics of C/E systems are modeled by a *discrete state realization* (defined in section 3), state transitions can be inhibited or enabled by the condition input signal and "forced" by the event input signal. The value of the condition output signal is a function of the system state and input condition signal, while the output events are a function of state transitions and input events. The state trajectory is a condition signal.

The primary motivation for introducing C/E systems is to provide an intuitive modeling framework for representing interconnected DEDS. In applications such as discrete control, condition and event signals characterize the flow of information between system components. A mixture of condition and event signals have also arisen in the DEDS literature in mechanisms proposed for interconnecting systems. This is illustrated by the interconnections between *plants* and *supervisors* defined by Ramadge and Wonham. For example, figure 2 illustrates the signal flow in the supervisory control structure proposed in reference [3] where events (state transitions) in the plant force simultaneous state transitions in the supervisor. Thus, an event signal flows from the plant to the supervisor in the C/E system model of this scheme while the supervisor controls the plant by a condition signal which enables and disables state transitions in the plant. In another paper, Ramadge and Wonham introduce a different supervisor structure which generates control inputs based on the state of the plant [4]. This feedback system is illustrated in figure 3 where the plant state is fed back as a condition signal to the supervisor which simply maps the the current state into the control input signal.

---

[1] Our use of the term condition/event systems is not related to the condition/event-systems defined by Reisig [1] as a class of Petri nets.

C/E systems also provide a framework for developing the relationships between continuous-time DEDS models used for simulation and performance evaluation, and strictly discrete models (such as automata, formal languages, and traces) used for qualitative analysis. Relationships between continuous-time C/E systems and the associated languages characterizing their discrete behavior are developed in [5]. In this paper we demonstrate that C/E systems with discrete state realizations provide a consistent modeling framework for representing interconnected C/E systems.

In the following section we define C/E signals and systems. In section 3 we consider C/E systems with discrete state realizations. These C/E systems are causal, time-invariant, and *spontaneous* in the sense that output events cannot be forced by transitions in the input condition signal. In section 4 we develop discrete state realizations for C/E systems resulting from cascade and feedback interconnections. The concluding section summarizes the results presented in this paper and discusses several directions for future research.

2 Condition/Event Signals and Systems

C/E systems are defined in terms of the following two classes of signals.

Definition 1. *Condition Signals.* Given a nonempty countable set X and an initial time $t_0 \in \mathbf{R}$, a function $x:[t_0, \infty) \rightarrow X$ is a *condition signal* over X on $[t_0, \infty)$ if $x(\bullet)$ is right continuous with limits from the left. The set of all condition signals over X on $[t_0, \infty)$ is denoted by $C(X,t_0)$.

Definition 2. *Event Signals.* Given a nonempty countable set X, a *null symbol* $\eta \notin X$, and an initial time $t_0 \in \mathbf{R}$, a function $x:[t_0, \infty) \rightarrow X \cup \{\eta\}$, is an *event signal* over X on $[t_0, \infty)$ if $x(t_0) = \eta$, and for any finite interval $[t_1, t_2] \subseteq [t_0, \infty)$ the set $\{t \in [t_1, t_2] \mid x(t) \neq \eta\}$ is finite. The set of all event signals over X on $[t_0, \infty)$ is denoted by $E(X,t_0)$.

Throughout the paper we use the null symbol η introduced in the definition of event signals to represent a null value which is not in any set of condition or event symbols. Given a countable set X, we define the notation $X^+ := X \cup \{ \eta \}$. In the definition of condition signals, and throughout the paper, right continuity is defined with respect to the discrete topology for countable sets in which all singleton sets are open. It is easily shown that condition signals have a finite number of discontinuities on a finite interval. Event signals represent discrete-event phenomena similar to point processes in stochastic settings. While the number of discontinuities in condition and event signals is always finite (but possibly unbounded) on any finite interval, they can have an infinite number of discontinuities on the infinite interval $[t_0, \infty)$.

Definition 3. *C/E System.* Given finite, disjoint sets U, V, Y, Z, and initial time $t_0 \in \mathbf{R}$, $S \subseteq C(U,t_0) \times E(V,t_0) \times C(Y,t_0) \times E(Z,t_0)$ is a *condition-event system* on $[t_0, \infty)$ with *input conditions* U, *input events* V, *output conditions* Y, and *output events* Z if for each *input* $(u(\bullet), v(\bullet)) \in C(U,t_0) \times E(V,t_0)$ there exists at least one *output* $(y(\bullet), z(\bullet)) \in C(Y,t_0) \times E(Z,t_0)$ such that $s(\bullet) = (u(\bullet), v(\bullet), y(\bullet), z(\bullet)) \in S$.

In words, a C/E system is simply a relation on the product of sets of condition and event input and output signals which identifies a set of output signals with each pair of input signals. Thus, C/E systems are in general nondeterministic. For brevity and simplicity of presentation, the sets U, V, Y, Z and the initial time t_0 will be assumed and invoked as needed when a general C/E system S is being discussed. We also use the symbol $s(\bullet)$ to represent a four-tuple of condition/event input-output signals in a system S. Properties of C/E systems, such as causality, time-change invariance and spontaneity, and classes of C/E systems, such as deterministic and discrete-time systems, are defined and illustrated in [5].

3 Discrete State Realizations

Models of C/E systems can involve signals that are not input or output signals, representing internal system dynamics. Willems calls such auxiliary signals *latent variables* [6]. *State variables* are the most common example of latent variables in systems theory. In this section we introduce a particular class of C/E models called *discrete state models* that are defined in terms of the C/E system input-output signals and a *state trajectory* which is a condition signal that takes values in a (possibly infinite) set of *discrete states.*

Definition 4. *Discrete State Model.* Given finite sets U, V, Y, Z, and an initial time t_0, a *discrete state model* Σ is a 5-tuple, $\Sigma = (X, f, g, h, X_0)$, where, X is a countable set of *states*; $f: X \times U \times V^+ \rightarrow \{2^X - \varnothing\}$ is the *state transition function* satisfying $x \in f(x, u, \eta)$, $\forall x \in X$, $\forall u \in U$; $g: X \times U \rightarrow Y$ is the *condition output function*; $h: X \times X \times V^+ \rightarrow Z^+$ is the *event output function* satisfying $h(x, x, \eta) = \eta$ for all $x \in X$; and $X_0 \subseteq X$ is a *set of initial states.* An input-output signal $s(\bullet) \in C(U,t_0) \times E(V,t_0) \times C(Y,t_0) \times E(Z,t_0)$ and a *state trajectory* $x(\bullet) \in C(X,t_0)$ satisfy the model Σ when $x(t_0) \in X_0$ and the following equations are satisfied for all $t > t_0$:

$$x(t) \in f(x(t^-), u(t^-), v(t))$$

$$y(t) = g(x(t), u(t))$$

$$z(t) = h(x(t^-), x(t), v(t)).$$

The output functions $g(\bullet, \bullet)$ and $h(\bullet, \bullet, \bullet)$ in discrete state models are analogous to output functions in state-variable models for continuous systems that permit the direct "feed-through" of input signals. Transitions in the output condition signal can be effected by input condition transitions as well as state transitions, and output events can result from input events as well as state transitions. The condition $h(x, x, \eta) = \eta$ for the event output function implies that a discrete state model cannot autonomously generate output events without some corresponding input event or internal state transition.

In [5] we demonstrate that discrete state models are in fact C/E models, in the sense that there always exists a solution to the discrete state model equations for an arbitrary input signal. Given a discrete state model Σ, the corresponding C/E system is denoted by $S(\Sigma)$. If S' is a C/E system and Σ is a discrete state model such that $S'=S(\Sigma)$, then Σ is called a *discrete state realization* of S'. It is easily shown that C/E systems with discrete state realizations are necessarily causal, time-scale invariant and spontaneous [5]. One implication of these results is that any inherent time dependence in a system behavior (that is, time dependence that is not due to input signals, including clocks), necessarily requires nondiscrete states, i.e., states with continuous components. For example, a state space realizations for C/E system that exhibits fixed delays in the internal dynamics must include a continuous state variable, representing the residence time in the current discrete "state".

4 Interconnected C/E Systems

One of the primary motivations for introducing discrete event systems with an explicit input/output signal structure is to permit clear, consistent definitions of interconnected systems. In this section we present the two fundamental interconnections for block diagram representations of systems, namely, *cascade* and *feedback* interconnections. The objective is to define a discrete state realization for the overall system, given discrete state realizations of the subsystems and a natural definition of the signal flow between subsystems.

4.1 Cascade Systems

Consider two C/E systems S_1, S_2 with discrete state realizations $\Sigma_1=(X_1,f_1,g_1,h_1,X_{1_0})$ and $\Sigma_2=(X_2,f_2,g_2,h_2,X_{2_0})$, respectively. In general, the cascade interconnection of system S_1 followed by S_2 requires a *condition connection mapping* $C:Y_1 \rightarrow U_2$ and an *event connection mapping* $E:Z_1 \rightarrow V_2$. The resulting cascade C/E system, represented by the notation $S(S_1 \rightarrow (C,E) \rightarrow S_2)$, is illustrated in figure 4. Without loss of generality, the mappings C and E can be absorbed into the output functions g_1 and h_1, respectively, for the discrete state realization Σ_1 for C/E system S_1, and it can be assumed that $Y_1 = U_2$ and $Z_1 = V_2$. The resulting cascade system, denoted by $S(S_1 \rightarrow S_2)$, is illustrated in figure 5.

The input-output behavior of $S(S_1 \rightarrow S_2)$ is given by the compositions of the state realizations for S_1 and S_2. In particular, the input-output behavior for $S(S_1 \rightarrow S_2)$ is all functions $(u_1(\bullet), v_1(\bullet), y_2(\bullet), z_2(\bullet))$ for which there are corresponding state trajectories $x_1(\bullet)$ and $x_2(\bullet)$ satisfying $x_1(t_0) \in X_{1_0}$, $x_2(t_0) \in X_{2_0}$, and the equations

$$x_1(t) \in f_1(x_1(t^-), u_1(t^-), v_1(t))$$

$$x_2(t) \in f_2(x_2(t^-), g_1(x_1(t^-), u_1(t^-)), h_1(x_1(t^-), x_1(t), v_1(t)))$$

$$y_2(t) = g_2(x_2(t), g_1(x_1(t), u_1(t)))$$

$$z_2(t) = h_2(x_2(t^-), x_2(t), h_1(x_1(t^-), x_1(t), v_1(t))),$$

where the intermediate signals $y_1(\bullet) = u_2(\bullet)$, $z_1(\bullet) = v_2(\bullet)$ have been eliminated.

Introducing the state space $X = X_1 \times X_2$ with a initial state set $X_0 = X_{1_0} \times X_{2_0}$, we can define output functions $g:X \times U_1 \rightarrow Y_2$ and $h:X \times X \times V_1^+ \rightarrow Z_2$ for a discrete state realization $\Sigma = (X,f,g,h,X_0)$ of the system $S(S_1 \rightarrow S_2)$ as:

$$g(x, u_1) = g_2(x_2, g_1(x_1, u_1)) \quad \text{for } x=(x_1,x_2) \in X,\ u_1 \in U_1$$

and

$$h(x,x',v_1) = h_2(x_2, x'_2, h_1(x_1, x'_1, v_1)) \quad \text{for } x=(x_1,x_2),\ x'=(x_1',x_2)' \in X \text{ and } v_1 \in V_1.$$

The state transition function $f:X \times U_1 \times V_1^+ \rightarrow 2^X$ cannot be defined as easily for $S(S_1 \rightarrow S_2)$, however, because of the dependence of the event function argument in f_2 on transitions in the state variable x_1 (due to the input of z_1 into S_2). To complete the discrete state realization $\Sigma=(X,f,g,h,X_0)$ such that $S(\Sigma) = S(S_1 \rightarrow S_2)$ we define

$$f(x,u_1,v_1) = \{x' = (x_1',x_2') \in X \mid x_1' \in f_1(x_1,u_1,v_1) \text{ and } x_2' \in f_2(x_2,g_1(x_1,u_1),h_1(x_1,x'_1,v_1))\}.$$

Note that this definition of f only permits transitions in the second component of the state in X_2 that are consistent with transitions in the first component of the state in X_1 that generate forcing events for S_2 via $v_2(t) = z_1(t) = h_1(x_1(t^-), x_1(t), v_1(t))$. Using the above definitions we have the following theorem.

Theorem 1. Given two systems S_1, S_2, with discrete state realizations Σ_1, Σ_2, and the cascade system $S(S_1 \rightarrow S_2)$, then the C/E system for the discrete state realization $\Sigma = (X, f, g, h, X_0)$ defined above is equivalent to $S(S_1 \rightarrow S_2)$, i.e. $S(\Sigma) = S(S_1 \rightarrow S_2)$.

Proof: Let $(s(\bullet), x(\bullet))$ solve the equations of Σ, where $s(\bullet) = (u_1(\bullet), v_1(\bullet), y_2(\bullet), z_2(\bullet)) \in S(\Sigma)$. Then $x(t_0) \in X_0$ and $\forall\, t \in (t_0, \infty)$, $x(t) \in f(x(t^-), u_1(t^-), v_1(t))$, $y_2(t) = g(x(t), u_1(t))$, and $z_2(t) = h(x(t^-), x(t), v(t))$. From the definitions of X_0, $f(\bullet, \bullet, \bullet)$, $g(\bullet, \bullet)$, and $h(\bullet, \bullet, \bullet)$, the components $x_1(\bullet)$, $x_2(\bullet)$ of $x(\bullet)$ satisfy the equations above for $S(S_1 \to S_2)$. Defining $y_1(\bullet) = u_2(\bullet) = g_1(x_1(t), u_1(t))$ and $z_1(\bullet) = v_2(\bullet) = h_1(x_1(t^-), x_1(t), v_1(t))$, it follows that $(u_1(\bullet), v_1(\bullet), y_1(\bullet), z_1(\bullet)) \in S_1$ and $(u_2(\bullet), v_2(\bullet), y_2(\bullet), z_2(\bullet)) \in S_2$. Hence $S(\Sigma) \subseteq S(S_1 \to S_2)$. The reverse inclusion follows similarly from the definitions of $f(\bullet, \bullet, \bullet)$, $g(\bullet, \bullet)$, and $h(\bullet, \bullet, \bullet)$.QED

4.2 Feedback Systems

Without loss of generality we consider the simple feedback structure shown in figure 6 where we assume the input and output sets are the same for the C/E system S in the feedback loop, that is, U = Y and V = Z. In general, such equivalence can be achieved with an appropriate pair of mappings from Y to U and Z to V, respectively. These mappings can simply be absorbed into the definitions of the output functions g and h in a discrete state realization for S. Furthermore, if a second C/E system is included in the feedback path, the structure of figure 6 can be obtained by simply applying the results for cascade systems from the previous section to obtain a discrete state realization for a single C/E system in the feedback loop. Thus, the C/E feedback system accommodates both feedback structures introduced by Ramadge and Wonham (figures 2 and 3).

Given a discrete state realization Σ for S, the feedback system resulting when U = Y and V = Z is the set of output signals $(y(\bullet), z(\bullet)) \in C(Y,t_0) \times E(Z,t_0)$ for which there exists an associated state trajectory $x(\bullet) \in C(X,t_0)$ such that $x(t_0) \in X_0$ and the following equations are satisfied for $t > t_0$:

$$x(t) \in f(x(t^-), y(t^-), z(t)),$$

$$y(t) = g(x(t), y(t)),$$

$$z(t) = h(x(t^-), x(t), z(t)).$$

Letting Σ_F denote the above equations, $S(\Sigma_F) \subseteq C(Y,t_0)\times E(Z,t_0)$ denotes the set of output signals $(y(\bullet), z(\bullet))$ for which there exists a corresponding state trajectory $x(\bullet) \in C(X,t_0)$ such that $(x(\bullet), y(\bullet), z(\bullet))$ satisfy the equations Σ_F. We note that the output equations impose constraints on the output signals which may not have any solutions, in which case $S(\Sigma_F)=\varnothing$. Observe that if the original system $\Sigma = (X, f, g, h, X_0)$ is such that $g{:}X \to Y$ and $h{:}X\times X \to Z^+$ (i.e., there is no direct "feed-through" of the input signals to the output signals), then the output equations for the feedback system Σ_F do not impose any constraints on the output signals and $S(\Sigma_F)$ is guaranteed to be non-empty.

Given the discrete state model Σ for the open loop system, we define the discrete state model $\Sigma' = (X', f', g', h', X'_0)$ for $S(\Sigma_F)$ as follows. $X' = X \times Y\times Z^+$, $X'_0 = \{x' = (x,y,z) \in X' \mid x = x_0,\ y = g(x_0, y), \text{ and } z = \eta\}$. It is shown below that the initial condition set X'_0 is nonempty if and only if $S(\Sigma_F)$ is nonempty. Letting $x'_1, x'_2 \in X'$ be arbitrary states with $x'_i = (x_i, y_i, z_i) \in X \times Y\times Z^+$, for i = 1, 2, the output functions $g'{:}X' \to Y$ and $h'{:}X' \times X' \to Z^+$ are defined by

$$g'(x'_1) = y_1$$

and

$$h'(x'_1, x'_2) = \begin{cases} z_2 & \text{if } z_2 \neq z_1 \\ \eta & \text{otherwise} \end{cases}$$

Finally, we define the state transition function $f':X' \to 2^{X'}$ as:

$$f'(x'_1) = \begin{cases} \Gamma'(x'_1) & \text{if } z_1=\eta \text{ and } \Gamma'(x'_1) \neq \emptyset \\ \{x'_1\} & \text{if } z_1=\eta \text{ and } \Gamma'(x'_1) = \emptyset \\ \{x'_1, (x_1, y_1, \eta)\} & \text{if } z_1 \neq \eta \end{cases}$$

where $\Gamma'(x'_1) = \{ x'_2 \in X' \mid y_2 = g(x_2, y_2), z_2 = h(x_1, x_2, z_2)$, and $x_2 \in f(x_1, y_1, z_2)\}$.

We note that Σ' as defined above is a discrete state model since $f'(x')$ is nonempty and $h'(x',x') = \eta$ for all $x' \in X'$ if Σ_F is non-empty. With these definitions we have the following result.

Theorem 2. Given a discrete state model Σ with U=Y and V=Z and an associated feedback model Σ_F, the discrete state model Σ' defined above is a discrete state realization for the C/E system $S(\Sigma_F)$, that is, $S(\Sigma') = S(\Sigma_F)$.

Proof: We consider two cases: (i) $S(\Sigma_F) = \emptyset$, and (ii) $S(\Sigma_F) \neq \emptyset$.

Case (i): We first observe $S(\Sigma_F)$ is empty if and only if X'_0 is empty. The "if" part of this observation is straightforward. The "only if" part is established by contradiction as follows. Suppose $S(\Sigma_F) = \emptyset$ and $X'_0 \neq \emptyset$, and choose $x'=(x,y,\eta) \in X'_0$. Then $y = g(x,y)$. Define the constant signals $y(t) = y$, $z(t) = \eta$, and $x(t) = x$, $\forall\, t \in [t_0, \infty)$. Clearly, $(y(\bullet), z(\bullet)) \in S(\Sigma_F)$, since $x(\bullet)$, $y(\bullet)$, $z(\bullet)$ trivially satisfy the equations of Σ_F, which contradicts the assumption that $S(\Sigma_F)$ is empty. So, if $S(\Sigma_F) = \emptyset$, then $X'_0 = \emptyset$, which in turn implies that Σ' has an empty initial state set. Therefore, $S(\Sigma') = \emptyset$, implying $S(\Sigma_F) = S(\Sigma')$.

Case (ii): Assume $S(\Sigma_F) \neq \emptyset$ and choose $s'(\bullet) = (y(\bullet), z(\bullet)) \in S(\Sigma')$. This implies there exists a state trajectory $x'(\bullet) \in C(X', t_0)$ such that $x'(t_0) \in X'_0$ and $(s'(\bullet), x'(\bullet))$ satisfy the discrete state model Σ', that is,

$$x'(t) \in f'(x'(t^-)),$$

$$y(t) = g'(x'(t)),$$

$$z(t) = h'(x'(t^-), x'(t)).$$

Let $x:[t_0, \infty) \to X$ be the condition signal $x(\bullet) \in C(X, t_0)$ resulting from the projection of $x'(t) \in X \times Y \times Z^+$ onto X for each $t \in [t_0, \infty)$. We claim that $(s'(\bullet), x(\bullet))$ satisfies the C/E model Σ_F. This follows from the following observations.

First, the definition of X'_0 guarantees that $x(t_0) \in X_0$ and $y(t_0) = g(x(t_0), y(t_0))$. The definition of $f'(\bullet)$ then guarantees that $y(t) = g(x(t),y(t))$ for any subsequent time t. We note that the definition of $g'(\bullet)$ implies $y(\bullet)$ is equal to the condition signal resulting from the projection of $x'(t) \in X \times Y \times Z^+$ onto Y for each $t \in [t_0, \infty)$.

Turning to the event output signal, let $z':[t_0, \infty) \to Z^+$ be the condition signal $z'(\bullet) \in C(Z^+, t_0)$ resulting from the projection of $x'(t) \in X \times Y \times Z^+$ onto Z^+ for each $t \in [t_0, \infty)$. The definition of $h'(\bullet, \bullet)$ implies that $z(t) = z'(t)$ at the points of discontinuity for $z'(\bullet)$. Moreover, $z(t) \neq \eta$ only when $z'(t^-) = \eta$ and $z'(t) \neq \eta$. The definition of $f'(\bullet)$ implies that $z'(t) = h(x(t^-), x(t), z'(t))$ whenever $z'(t^-) = \eta$ and $z'(t) \neq \eta$. Therefore, $z(t) = h(x(t^-), x(t), z(t))$ for all t such that $z(t) \neq \eta$. On the other hand, suppose that for some t, $z(t) = \eta$ but $h(x(t^-), x(t), \eta) \neq \eta$. This implies $x(t^-) \neq x(t)$. From the definition of $f'(\bullet)$, $x(\bullet)$ can be discontinuous at a time t only when $z'(t) = h(x(t^-), x(t), z'(t))$ and $z'(t^-) = \eta$. Since we are considering the case when $z(t) = \eta$, $z'(t) = \eta$, which contradicts the assumption that $h(x(t^-), x(t), \eta) \neq \eta$.

Finally, turning to the state transition equation, we have from the above discussion that $z(t) = \eta$ except when $z'(t^-) = \eta$ and $z'(t) \neq \eta$, in which case $z(t) = z'(t)$. This fact, along with the definition of $f'(\bullet)$ and the general property that $x \in f(x, u, \eta)$, imply that $x(t) \in f(x(t^-), y(t^-),$

z(t)). Therefore, (s′(•), x(•)) satisfies Σ_F, implying s′(•) ∈ S(Σ_F), which in turn implies S(Σ′) ⊆ S(Σ_F).

Now, suppose s(•) = (y(•), z(•)) ∈ S(Σ_F). This implies there exists a condition signal x(•) ∈ C(X, t_0) such that x(t_0) ∈ X_0 and (s(•), x(•)) satisfy the equations Σ_F, that is,

$$x(t) \in f(x(t^-), y(t^-), z(t))$$

$$y(t) = g(x(t), y(t))$$

$$z(t) = h(x(t^-), x(t), z(t)).$$

To show s(•) ∈ S(Σ′), it is necessary to construct a condition signal (state trajectory) x′(•) ∈ C(X′, t_0) such that x′(t_0) ∈ X'_0 and (s(•), x′(•)) satisfy the discrete state model Σ′. We construct such a state trajectory as follows. Let $t_k = t_k[z(\bullet)]$, k= 0, 1, ... denote the points of discontinuity in z(•). For each t_k (when it is defined) choose $\varepsilon_k > 0$ such that y(•) and x(•) are constant on the interval $[t_k, t_k+\varepsilon_k]$. (Such ε_k exist since y(•) and x(•) are condition signals.) Define the condition signal z′(•) ∈ C(Z^+,t_0) as

$$z'(t) = \begin{cases} z(t_k) & t_k \le t < t_k+\varepsilon_k \\ \eta & \text{otherwise} \end{cases}$$

We claim the condition signal $x':[t_0, \infty) \to X'$ defined at each t ∈ $[t_0, \infty)$ by x′(t) = (x(t), y(t), z′(t)) is one of the desired state trajectories. This is easily verified from the definitions of X'_0, f′(•), g′(•), and h′(•, •). Therefore, s(•) ∈ S(Σ′), which in turn implies S(Σ_F) ⊆ S(Σ′).QED

To illustrate the feedback connection and its discrete state realization, consider the identity system defined by the input/output equations y(t) = u(t) and z(t) = v(t). Connecting the outputs to inputs for this system yields a feedback system characterized by the trivial equations y(t) = y(t) and z(t) = z(t). Thus, the feedback system is simply all possible output signals, C(Y,t_0)×E(Z,t_0). A discrete state realization Σ′ = (X′, f′, g′, h′, X'_0) for this system is defined by the functions f′, g′, h′ given above with the state set X′ = Y ×Z^+, the initial condition set X'_0 = Y×{η}. The state set X is not involved in this case since the open loop system has no dynamics. Note, however, that the discrete state realization of the feedback system does have dynamics: state transitions are required to generate the event output signal z(•) from the autonomous system.

5 Discussion

This paper introduces C/E systems with discrete state realizations. Discrete state realizations are derived for cascade and feedback C/E systems. The definitions of interconnected C/E systems easily and intuitively model causality in the flow of signals between subsystems, particularly with respect to forced state transitions. Forced transitions have been considered using formal language models by Golaszewski and Ramadge [7], and using trace theoretic models by Inan [8] and Heymann [9]. The difference in our approach is that the input-output model provides a continuous-time signal-based representation of situations in which state transitions (events) in one system (S_1) are independent of state transitions in another system (S_2), but state transitions in S_2 may be contingent on events in S_1. Heymann has modeled such behaviors using the concept of *prioritized events.*

There are several directions for future research in C/E systems. Standard system theoretic issues to be addressed include the definition and existence of minimal discrete state realizations, canonical forms, and suitable definitions of observability and controllability. Relationships between the structural properties of certain C/E models and the input-output behaviors of the corresponding C/E systems are currently under investigation. The C/E systems framework admits the possibility for formulating and solving a larger class of feedback control problems than has been considered to date in the discrete event systems literature.

C/E systems also provide a framework for extending results developed using other modeling formalisms such as formal languages, trace theory, Petri nets, and automata, to include explicit input-output signaling. Discrete state realizations developed in the present paper are essentially an extension of an automata-based approach. Petri net models for the internal dynamics of C/E systems have been considered in [5]. Since condition and event signals correspond to physical signals in applications, it is hoped that methods for C/E systems analysis and synthesis could ultimately provide the much needed link between theory and practice in the discrete control systems.

Acknowledgments

This research was sponsored in part by the National Science foundation under grant number DMC-8451493 and grants from the Philips Corporation and Rockwell International.

References

1. W. Reisig, *Petri Nets: An Introduction,* Springer-Verlag, Berlin, 1982.
2. S. Bennett, *Real-Time Computer Control: An Introduction,* Prentice Hall, New York, 1988.
3. P.J. Ramadge and W.M. Wonham, "Supervisory control of a class of discrete event processes", *SIAM J. Control and Optimization*, Vol. 25, No. 1, January 1987, pp. 206-230.
4. P.J. Ramadge and W.M. Wonham, "Modular feedback logic for discrete event systems", *SIAM J. Control and Optimization*, Vol. 25, No. 5, September 1987, pp. 1202-1218.
5. R.S. Sreenivas, *Towards a system theory for interconnected condition/event systems,* PhD dissertation, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, Sept. 1990, Tech. Report No. LASIP 90-008.
6. J.C. Willems, "Models for Dynamics", Tech. report, Systems and Control Group, Mathematics Institute, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands, July 1987.
7. C.H. Golaszewski and P.J. Ramadge, "Control of Discrete Event Processes with Forced Events", *Proc. of the 26th Conference on Decision and Control*, IEEE, Los Angeles, CA, December 1987, pp. 247-251.
8. K. Inan, "Asynchronous Dynamical Systems", *Proceedings of the 28th Conference on Decision and Control*, Tampa, Florida, December 1989, pp. 1544-1549.
9. M. Heymann, "Concurrency and discrete event control", *IEEE Control Systems Magazine*, Vol. 10, No. 4, June 1990, pp. 103-112.

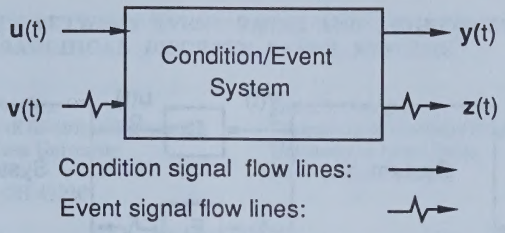

u(t)
v(t)
Condition/Event System
y(t)
z(t)
Condition signal flow lines:
Event signal flow lines:

Figure 1: Basic input/output signal flow for a C/E system.

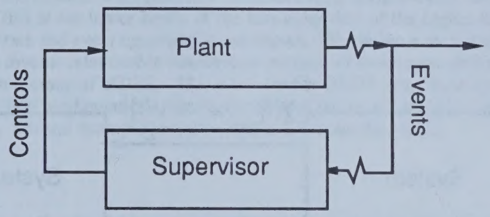

Plant
Controls
Events
Supervisor

Figure 2: Signal flow for the supervisory controller proposed in [3].

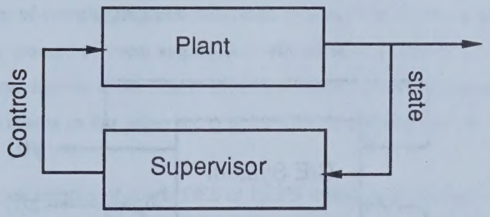

Plant
Controls
state
Supervisor

Figure 3: Signal flow for the supervisory controller proposed in [4].

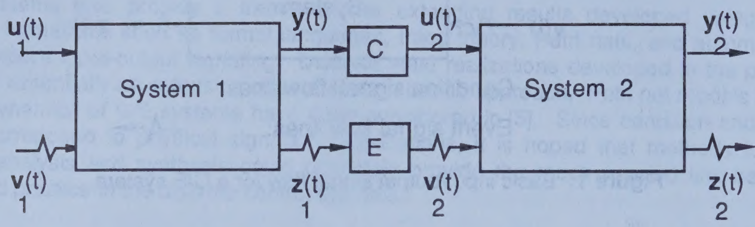

u(t) 1
v(t) 1
System 1
y(t) 1
z(t) 1
C
E
u(t) 2
v(t) 2
System 2
y(t) 2
z(t) 2

Figure 4: Cascade interconnection of two C/E systems.

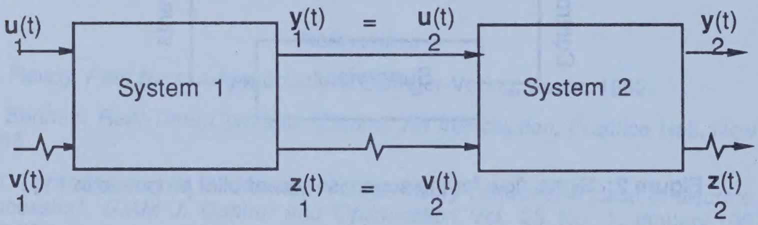

u(t) 1
v(t) 1
System 1
y(t) 1 = u(t) 2
z(t) 1 = v(t) 2
System 2
y(t) 2
z(t) 2

Figure 5: Simplified cascade interconnection of two C/E systems.

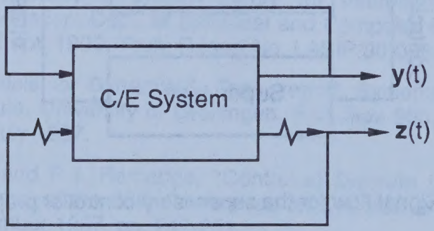

C/E System
y(t)
z(t)

Figure 6: A feedback connection on a C/E system.

RELATIONSHIPS BETWEEN EVENT RATES AND AGGREGATION IN HIERARCHICAL DISCRETE EVENT SYSTEMS*

Kevin M. Passino
Department of Electrical Engineering
The Ohio State University
2015 Neil Ave.
Columbus, OH 43210

Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

A discrete event system (DES) is a dynamical system whose evolution in time develops as the result of the occurrence of physical events at possibly irregular time intervals. Although many DES's operation is asynchronous, others have dynamics which depend on a clock or some other complex timing schedule. Here we provide a formal representation of the advancement of time for logical DES via *interpretations of time*. We show that the interpretations of time along with a *timing structure* provide a framework to study principles of the advancement of time for hierarchical DES (HDES). In particular, it is shown that for a wide class of HDES the *event rate* is higher for DES at the lower levels of the hierarchy than at the higher levels of the hierarchy. Relationships between event rate and *event aggregation* are shown. We define a measure for event aggregation and show that there exists an inverse relationship between the amount of event aggregation and the event rate at any two successive levels in a class of HDES. These results for HDES constitute the main results of this paper since they provide the first mathematical characterization of the relationship between event aggregation and event rates of the HDES. Several examples are provided to illustrate the results.

1.0 INTRODUCTION

In our main results we show that the *interpretations of time* which characterize the advancement of time in DES (introduced in Section 2) along with a *timing structure* provide a framework to study principles of the advancement of time for hierarchical DES (HDES). In Section 3 it is shown that for a wide class of HDES the *event rate* is higher for DES at the lower levels of the hierarchy than at the higher levels of the hierarchy. Relationships between event rate and *event aggregation* are shown. We define a measure for event aggregation and show that a high amount of event aggregation will result in a much lower event rate at higher levels in a class of HDES while a low amount of event aggregation will result in higher event rates. The results are illustrated with a conventional discrete event control system, a hybrid dynamical system and a manufacturing system. The proofs for the results in this paper are contained in [8] and some of the results were originally reported in [4,6,9].

We focus on timing characteristics of single DES or HDES which have as components DES that can be accurately modelled with

$$P=(X,U,Y,\delta,\lambda,X_0) \qquad (1)$$

where,

(i) X is the set of plant states x,

(ii) U is the set of plant inputs u,

(iii) Y is the set of plant outputs y,

(iv) $\delta:U\times X\rightarrow\mathbb{P}(X)$ is the plant state transition function ($\mathbb{P}(X)$ denotes the power set of X),

(v) $\lambda:U\times X\rightarrow Y$ is the plant output function, and

(vi) $X_0\subset X$ is the set of possible initial plant states.

---

* The authors gratefully acknowledge the partial support of the Jet Propulsion Laboratory. Please address all correspondence to K. Passino.

The plant state transition function (a partial, point to set function) specifies for each current input u and state x the set of possible next states $x' \in \delta(u,x)$. The output function specifies, for the current input u and state x, the current output symbol $y=\lambda(u,x)$. Formally, P is equivalent to a directed graph with node set X and edges $x \rightarrow x'$ labelled with "u/y" for each triple (u,x,x') such that $x' \in \delta(u,x)$ and $y=\lambda(u,x)$. The model P is similar to a standard automaton but X, U, and Y are not required to be finite. A *run* of P is defined as a sequence of triples (u_0,x_0,y_0), (u_1,x_1,y_1), (u_2,x_2,y_2), ... such that $x_0 \in X_0$, u_0 is the initial input, $x_{k+1} \in \delta(u_k,x_k)$, and $y_k=\lambda(u_k,x_k)$. Notice that since it is possible that $\delta(u_k,x_k)=\emptyset$ for all $u_k \in U$ at some $x_k \in X$, a run may have a finite length. The model P can be classified as being a "logical DES model" since, often it is used to study logical DES properties.

2.0 CHARACTERIZING THE ADVANCEMENT OF TIME IN DES

When a physical plant is modelled via (1), the meaning of the advancement of time must be defined. If Z is an arbitrary set, then Z^* denotes the set of all finite strings of elements from Z. If Z and Z' are arbitrary sets then $Z^{Z'}$ denotes the set of all functions mapping Z' to Z. In order to discuss timing issues for P, an *index set* J and *index sequences*

$$\alpha \in J^* \cup J^{\mathbb{N}}$$

are utilized similar to the approach in [12] where $\mathbb{N}$ denotes the set of natural numbers. The index set J is thought of as a set of times. Let $\mathbb{R}^+$ denote the set of strictly positive real numbers and $\mathbb{R}_+ = \mathbb{R}^+ \cup \{0\}$, the set of non-negative reals. Note that $\mathbb{N}$, $\mathbb{R}_+$, or $\mathbb{R}$ could be candidates for the set J. The index sequences $\alpha \in J^* \cup J^{\mathbb{N}}$ are sequences of time instants that can be of finite or infinite length. For $\alpha \in J^* \cup J^{\mathbb{N}}$ let $|\alpha|$ denote the cardinality of the size of the string α. Note that either $\alpha: \mathbb{N} \rightarrow J$ or $\alpha:[0,a] \rightarrow J$ where $[0,a] \subset \mathbb{N}$, and $\alpha(k)$ simply denotes an element in J. An index sequence (function) $\alpha \in J^* \cup J^{\mathbb{N}}$ is said to be *admissible* if

(i) it is order preserving, i.e.,

(a) if $\alpha \in J^{\mathbb{N}}$, then for all $k_1,k_2 \in \mathbb{N}$, $k_1 \leq k_2$ implies that $\alpha(k_1) \leq \alpha(k_2)$,

(b) if $\alpha \in J^*$, then for all $k_1,k_2 \in \mathbb{N}$ with $k_1,k_2 \in [0,|\alpha|-1]$, $k_1 \leq k_2$ implies that $\alpha(k_1) \leq \alpha(k_2)$, and

(ii) it is injective.

Following [12], the state of the plant $x \in X$ is associated with the index $\alpha(k)$ for some $\alpha \in J^* \cup J^{\mathbb{N}}$ and is denoted with $x(\alpha(k))$, meaning "the state at time $\alpha(k)$". Similarly, inputs $u \in U$ and outputs $y \in Y$ are associated with that same index and denoted with $u(\alpha(k))$ and $y(\alpha(k))$ respectively. The transition to a state in the set $\delta(u,x)$ can be thought of as leading to the next state, with "next" quantified with the index sequence α as $\alpha(k+1)$. With this, the transition function is given as $x(\alpha(k+1)) \in \delta(u(\alpha(k)),x(\alpha(k)))$ which is often abbreviated as $x_{k+1} \in \delta(u_k,x_k)$. Similarly, the output is often denoted with $y_k=\lambda(u_k,x_k)$ for $k \in \mathbb{N}$. Each run of P $(u_0,x_0,y_0),(u_1,x_1,y_1),\ldots$ has an associated index sequence $\alpha \in J^* \cup J^{\mathbb{N}}$, $\alpha=\alpha(0),\alpha(1),\ldots$ specifying the time instants at which the triples are defined. Notice that if for some $x_k \in X$ and all $u_k \in U$, $\delta(u_k,x_k)=\emptyset$, then $\alpha(k+1)$ is undefined and in this case α has finite length (in this situation we say that P is "deadlocked").

A DES often *activates* or *triggers* other DES to act. For instance, in the case where P represents a plant, P may trigger a controller to generate an input to P. In this case, the trigger often represents certain changes that occur in the plant. For instance, *events* can be used as the trigger. Similar to [11], we let $E \subset X \times X$ denote the set of *events* e, where

$$E=\{(x,x')\in X\times X:\ x'\in \delta(u,x)\} \tag{2}$$

An event $e=(x,x')$ is said to *occur* if the state transition from x to $x'\in \delta(u,x)$ takes place. For convenience we shall assume that the event occurs (is defined) at the time instant $\alpha(k+1)$ where the next state is defined.

The pair $I=(A,J)$ where J is an index set and $A\subset J^{*}\cup J^{\mathbb{N}}$ will be referred to as an *interpretation of time* since it specifies the meaning of the advances in time for the occurrence of state transitions, i.e. it specifies the time instants where the variables of the DES P are defined. In general, a system P is said to have a particular interpretation of time $I=(A,J)$ as long as the time instants associated with the elements of the runs of P are elements of J and the index sequences associated with the runs of P are elements of A. The *admissible interpretation of time* will be denoted with $I_{ad}=(A_{ad},J_{ad})$ where J_{ad} is an index set and

$$A_{ad}=\{\alpha\in J_{ad}^{*}\cup J_{ad}^{\mathbb{N}} : \alpha \text{ is admissible}\}. \tag{3}$$

Most often we can choose $J_{ad}=\mathbb{R}_{+}$ and this is what we will assume here. It is common to discuss the timing characteristics of DES relative to a *clock*. By a "clock" we mean a device which has a fixed interval $T\in \mathbb{R}^{+}$ between *ticks* and which does not stop ticking (if there is deadlock, the clock keeps ticking but no events occur). Next we provide definitions for several standard interpretations of time:

Definition 1: The *asynchronous interpretation of time* is $I_a=(A_a,J_a)$ where $J_a=\mathbb{R}_{+}$ and
$A_a=\{\alpha\in A_{ad}: \alpha(0)=0\}$.

Definition 2: The *partially asynchronous interpretation of time* is $I_{pa}=(A_{\gamma\beta},J_{pa})$ with $J_{pa}=\mathbb{R}_{+}$
and $A_{\gamma\beta}=\{\alpha\in A_a : \alpha(k)+\gamma\leq\alpha(k+1)\leq\alpha(k)+\beta\}$ for $\gamma,\beta\in \mathbb{R}^{+}$ where $\beta\geq\gamma$.

Definition 3: The *general synchronous interpretation of time* is $I_s=(A_T,J_s)$ with $J_s=\mathbb{R}_{+}$
and $A_T=\{\alpha \in A_a : \alpha(k+1)=\alpha(k)+nT \text{ where } n\in \mathbb{N}-\{0\}\}$ with $T\in \mathbb{R}^{+}$.

When n=1 we shall refer to I_s simply as the *synchronous interpretation of time*.

3.0 TIMING CHARACTERISTICS OF HIERARCHICAL DES

The formation of a control theory for HDES is just beginning [15-17] even though such systems occur quite frequently. Some principles of the evolution of time in hierarchical systems have been postulated but not fully investigated in [1,13,14,5,7,2]. As in [3] what these researchers have recognized is that "systems usually operate at the higher rates at the lower levels in a hierarchical system". We shall verify this intuition for a wide class of HDES here.

3.1 A Hierarchical DES Model

We shall focus on HDES that have as components two types of DES, G_j, $1\leq j\leq m$, and P_i, $1\leq i\leq n$, all defined via (1) except with different timing characteristics. We introduce what we call a *timing structure* which will define how the various components of the HDES influence (are influenced by) the timing characteristics of other components of the HDES. The definition of the timing structure is based on the interpretations of time defined in Section 2.0 and what will be called *input* and *output triggers*. Each P_i, $1\leq i\leq n$, in the HDES has timing characteristics that are simply specified via their own interpretation of time denoted with $I_{pi}=(A_{pi},J_{pi})$.

Roughly speaking, each G_j, $1 \leq j \leq m$, has timing characteristics that depend on other P_i and G_ℓ via the timing structure as we now discuss in more detail.

Let E_{pi} denote the set of events for P_i, and E_{gj}, the set of events for G_j both defined in a similar manner to the events E for P in (2). Let C_{pi} (C_{gj}) denote the set of *communications* that can be transmitted from P_i (G_j) via the timing structure to other G_ℓ. The *output triggers* for the P_i are defined via

$$\phi_i : E_{pi}^* \rightarrow C_{pi} \quad 1 \leq i \leq n \tag{4}$$

and for the G_j by

$$\psi_j : E_{gj}^* \rightarrow C_{gj} \quad 1 \leq j \leq m \tag{5}$$

(or restrictions of these maps). If $\phi_i(e)=c$ or $\psi_j(e')=c'$ then c and c' are communications that are said to *occur* due to the occurrence of event string $\mathbf{e}$ or $\mathbf{e}'$ ($\mathbf{e}$ triggers communication c). We use the standard notation for *concatenation*, e.g., if $\mathbf{e}, \mathbf{e}' \in E_{pi}^*$ then $\mathbf{e}\mathbf{e}'$ denotes the concatenation of $\mathbf{e}$ and $\mathbf{e}'$. The time instant at which the communication c (c') occurs is the same time instant that $e \in E_{pi}^*$ ($e' \in E_{gj}^*$) occurs where $\phi_i(\mathbf{e}e)=c$ ($\psi_j(\mathbf{e}'e')=c'$). The *input triggers* for the G_j are defined by the τ_j maps for j, $1 \leq j \leq m$, where

$$\tau_j : C_{p1} \times \cdots \times C_{pn} \times C_{g1} \times \cdots \times C_{gj-1} \times C_{gj+1} \times \cdots \times C_{gm} \rightarrow \{0,1\} \tag{6}$$

and $\tau_j(\cdot)=1$ ($=0$) indicates that an event $e_{gj}(\alpha(k+1)) \in E_{gj}$ where $e_{gj}(\alpha(k+1))=(x_{gj}(\alpha(k)),x_{gj}(\alpha(k+1)))$ is forced (not) to occur in G_j. The τ_j indicate which P_i and G_ℓ communicate with G_j via the timing structure. Equation 6 indicates the form for the τ_j maps; the absence of a C_{pi} or $C_{g\ell}$ from the cross product in the domain of τ_j indicates that P_i or G_ℓ does not communicate with G_j via the timing structure. It is assumed that the τ_j maps form a "tree structured" timing structure as we describe next.

Let each DES component P_i or G_ℓ of the HDES represent a *node* (e.g., denoted with boxes as in Figure 3.1) of a directed graph $\mathcal{G}$ and let the τ_j define the *arcs* (e.g., denoted with shaded arcs in Figure 3.1) that connect the P_i and G_ℓ to G_j in the following manner: Let $i1, i2, \ell 1, \ell 2 \in \mathbb{N}$. If there exists i ($\ell$) such that $\tau_j : C_{pi1} \times \cdots \times C_{pi} \times \cdots C_{pi2} \times C_{g\ell 1} \times \cdots \times C_{g\ell} \times \cdots \times C_{g\ell 2} \rightarrow \{0,1\}$ then there exists an arc pointing from P_i to G_j (G_ℓ to G_j). In this paper we focus on HDES which have a *tree structured* timing structure, i.e., the case where $\mathcal{G}$ has no *closed cycles*. In this way we eliminate the possibility that some G_j can directly force its own events to occur via the timing structure. Notice that the P_i, $1 \leq i \leq n$, are the "leaves" of the tree structured timing structure.

Intuitively, the ϕ_i and ψ_j specify what each DES will communicate (C_{pi} and C_{gj}) to the other DES in the HDES. The τ_j define *communication channels* (the arcs and *paths* in $\mathcal{G}$) and where the communications are distributed in the HDES. Next, we define the time instants at which events occur when they are forced to do so by other DES components of the HDES via the timing strucutre.

Whereas the interpretation of time is always specified for the P_i, $1 \leq i \leq n$, the interpretations of time for the G_j are specified in terms of the other G_ℓ and the P_i via the timing structure. Let $\alpha_{cpi}(k+1)$ and $\alpha_{cg\ell}(k+1)$ denote the time instants at which communications $c_{pi} \in C_{pi}$ and $c_{g\ell} \in C_{g\ell}$ occur respectively. Suppose that at some time instant $\alpha_{gj}(k+1)$, $\tau_j(\cdot)=1$ so that $e_{gj}(\alpha_{gj}(k+1)) \in E_{gj}$ occurs. This time instant at which $e_{gj}(\alpha_{gj}(k+1))$ occurs is given by

$$\alpha_{gj}(k+1)=\max\{\alpha_{cpi}(k+1),\alpha_{cg\ell}(k+1) : 1 \leq i \leq n, 1 \leq \ell \leq m, \text{and } \exists \text{ an arc in } \mathcal{G} \text{ from } P_i \text{ or } G_\ell \text{ to } G_j\} \tag{7}$$

and corresponds to the time instant at which the last communication accesible to G_j occurred and caused $\tau_j(\cdot)=1$. Each time a communication occurs which forces $\tau_j(\cdot)=1$, an event occurs in G_j; hence the "1" represents a pulse sent to G_j via τ_j which forces an event to occur. Hence, if $\tau_j(\cdot)$ is set equal to 1 at some time instant, an event in G_j must occur at that time instant (unless G_j is deadlocked); if every communication in a sequence of communications all cause $\tau_j(\cdot)=1$ then there is one event occurrence in G_j for each communication in the sequence. The interpretation of time for any G_j is found by executing all possible runs (in all possible orders) of the P_i, $1 \leq i \leq n$, and G_ℓ for which there exists a path in $\mathcal{S}$ from P_i or G_ℓ to G_j. Then via (7), the time instants and hence index sequences and interpretations of time for the G_j are specified. We shall study HDES where there is at least one P_i and the interpretations of time for the G_j can be uniquely defined in terms of the P_i.

Note that although we consider only tree structured timing structures we place no restrictions on the manner in which the DES inputs and outputs are connected. This allows our results to apply to a large class of HDES with a wide variety of input/output connecting structures. Tree structured timing structures allow us to study properties of what has been called a "time scale hierarchy". In this hierarchy a DES component is "higher in the hierarchy" than another DES component if its timing characteristics can be influenced by the other DES (i.e., there exists a path in $\mathcal{S}$ from one to the other).

3.2 Lower Event Rates at Higher Levels in the HDES

To analyze the timing characteristics of HDES we study one fundamental component (shown in Figure 3.1) of the HDES defined above. Even though we consider only P_i at the lower level, it requires only a simple modification to consider a mix of P_i and G_j at the lower level and all of our results below are still valid. Moreover, our results easily generalize to the fully interconnected HDES by repeated application of the derived relationships which pertain to the two levels in Figure 3.1.

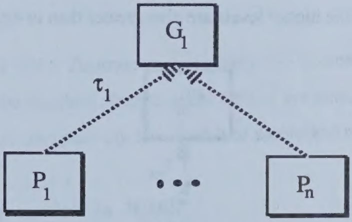


Figure 3.1 Hierarchical DES with Single-Branch

Let the admissible interpretation of time for P_i be $I_{pi}=(A_{pi},J_{pi})$, $1 \leq i \leq n$, with $J_{pi}=\mathbb{R}_+$ and for G_1 be $I_{g1}=(A_{g1},J_{g1})$. For any possible runs made by the P_i with an index sequence $\alpha_{pi} \in A_{pi}$, the corresponding run in G_1 have index sequence denoted by $\alpha_{g1} \in A_{g1}$.

Definition 4: The *event occurrence rate* (*event rate*) in P_i or G_j is the number of events that occur in the time interval $T_u=(r_1,r_2]$ where $(r_1,r_2] \subset \mathbb{R}^+$ and it will be denoted with $\#(P_i,T_u)$ and $\#(G_j,T_u)$ respectively.

Notice that if P_i has a synchronous interpretation of time with $T \in \mathbb{R}^+$ and we choose T_u such that $|r_2-r_1|=T$ then $\#(P_i,T_u)=1$, i.e., there is 1 event occurrence in the time interval T_u no matter what the particular values of r_1 and r_2 are. If P_i has an asynchronous interpretation of time then no matter how T_u is chosen it is possible that $\#(P_i,T_u)=0$, since we cannot guarantee that an event will occur in the given time interval T_u. In fact, we do not know how many events will occur in T_u. It would appear that our definition of event rate is too restrictive. This is, however, not the case since the focus here is on *comparing* the event rates of different DES components in the HDES and this comparison is made relative to T_u, an interval of the real time line.

Theorem 1: $\sum_{i=1}^{n}\#(P_i,T_u)\geq\#(G_1,T_u)\geq 0$ for all T_u.

Proof: Suppose that an event $e\in E_{pi}$ occurs at time $\alpha_{pi}(k)$ in some P_i, $1\leq i\leq n$ resulting in $\tau_1(\cdot,\cdot,\ldots,\phi_i(ee),\ldots,\cdot,\cdot)=1$. If another event $e'\in E_{p\ell}$ occurs in some P_ℓ at time $\alpha_{p\ell}(k')$ resulting in $\tau_1(\cdot,\cdot,\ldots,\phi_i(e'e'),\ldots,\cdot,\cdot)=0$ then the index sequence α_{g1} will contain $\alpha_{pi}(k)$ but not $\alpha_{g\ell}(k')$. Hence, α_{g1} is a subsequence of any α_{pi}, $1\leq i\leq n$, so clearly for any T_u the relationship holds (even if there is deadlock in any P_i or simlutaneous events occuring in any number of P_i). ∎

Theorem 1 states the intuitively clear fact that the timing structure can *mask* events and hence remove the time instants at which events occur in higher levels of the hierarchy. This means that the event rate is lower in DES at the higher levels of the HDES and higher in lower levels of the HDES no matter what the interpretations of time are for the P_i, $1\leq i\leq n$.

Remark 1: Repeated application of Theorem 1 to the *multi-level hierarchy* in Figure 3.2 results in $\#(P_1,T_u)\geq\#(G_1,T_u)\geq\cdots\geq\#(G_m,T_u)\geq 0$ for all T_u. This result supports the studies in [3] where the author assumes a synchronous interpretation of time and that the event rates can be split into "spectra" according to the level in the hierarchy. The result also shows that in the more general case, for an asynchronous P_1 for instance, the event rates in DES at the higher levels are also greater than or equal to the event rates at the lower levels.

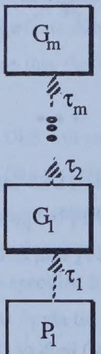


Figure 3.2 Multi-Level Hierarchical DES with m+1 Levels

Example 1: *(Conventional Discrete Event Control System)* Consider the controlled DES shown in Figure 3.3. We have $\phi_1:E^*_{p1}\rightarrow C_{p1}$ and $\tau_1:C_{p1}\rightarrow\{0,1\}$ and for the standard control configuration it is most often assumed that for all $e\in E^*_{p1}$ such that $e=e'e$ $(e\in E_{p1})$, $\tau_1(\phi_1(e'e))=1$ so that each time an event occurs in P_1, G_1 is forced to act by having an event in G_1 occur (it is normally assumed that one always exists). Clearly, then if $I_{p1}=(A_{p1},J_{p1})$ is the interpretation of time for P_1 and $I_{g1}=(A_{g1},J_{g1})$ for G_1 where $J_{g1}=J_{p1}$, then $A_{g1}=A_{p1}$. The interpretation of time for the plant and controller are the same. In this way we think of specifying the interpretation of time for G_1 by I_{p1} and P_1 via τ_1 and ϕ_1. Intuitively, for general ϕ_1 and τ_1 we see that we can expect fewer events to occur in G_1 than in P_1 since P_1 may not communicate the occurrence of an event or G_1 may not recognize the communication. ∎

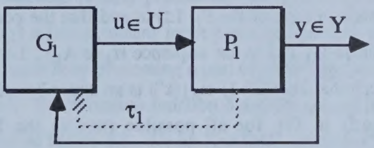


Figure 3.3 Discrete Event Control System

Example 2: *(Hybrid Dynamical System)* Let n=1 in Figure 3.1 and for $P_1=(X_p,U_p,Y_p,\delta_p,\lambda_p,X_{0p})$ let $X_p=\mathbb{R}^{np}$, $U_p=\mathbb{R}^{mp}$, $Y_p=\mathbb{R}^{rp}$, and I_{p1} be an admissible interpretation of time so that P_1 is a model for a nonlinear discrete time system. For instance P_1 could represent a zero-order-hold followed by a nonlinear continuous time system and a sampler. We think of G_1 as our DES model. For instance G_1 can model any system that can be represented by a General or Extended Petri net [10]. P_1 and G_1 constitute a hybrid dynamical system. Theorem 1 shows that if the event rate for P_1 is the number of state transitions per unit time then the event rate in the higher level DES model G_1 will be lower no matter how the communications are defined. ∎

Remark 1 and Examples 1 and 2 illustrate the generality of Theorem 1; the result applies to hierarchical DES currently being studied, the standard discrete event control systems, and to hybrid dynamical systems - a class of system for which there exist relatively few models or theoretical results.

3.3 Event Rates and Aggregation in HDES

Next we study how aggregation affects the event rates in HDES. Again we shall focus on the HDES component shown in Figure 3.1 but note that the result easily generalizes to a fully interconnected HDES. Let $E^*_{ai}\subset E^*_{pi}$ and $\phi_i':E^*_{ai}\rightarrow C_{pi}$ for all i, $1\le i\le n$, denote restrictions of ϕ_i. We use ϕ_i' maps for aggregation rather than the ϕ_i; if $e\in E^*_{pi}$, $e\notin E^*_{ai}$ then ϕ_i is said to *ignore* e (rather than *mask* e). Let $B_j\subset\mathbb{N}-\{0\}$ for j, $1\le j\le n$.

Definition 5: $\{P_j=(X^j,U^j,Y^j,\delta^j,\lambda^j,X_{0j}),\ \phi'_j:1\le j\le n\}$ satisfies the *$(\pi^1,\pi^2,\ldots,\pi^n)$-event aggregation property* if for each j, $1\le j\le n$,

(i) There exists a family of sets $X_{ij}\subset X^j$, $i\in B_j$ such that

(a) $X_{ij}\cap X_{kj}=\emptyset$ for all $i\neq k$, and $X_{0j}\cap X_{ij}=\emptyset$ for $i\in B_j$,

(b) If P_j first enters a state $x\in X_{ij}$ for some $i\in B_j$, it will take (for all possible runs) at least $\pi^j>0$ state transitions before the state of P_j, say x', is such that $x'\notin X_{ij}$,

(ii) $\phi'_j: E^*_{aj}\to C_{pj}$ where $E^*_{aj}=\{e\in E^*_{pj}: e=e'e$, where $e=(x,x')$ and for some $i\in B_j$, $x\in X_{ij}$, $x'\notin X_{ij}\}$.

Theorem 2: If $\{P_j=(X^j,U^j,Y^j,\delta^j,\lambda^j,X_{0j}), \phi'_j: 1\le j\le n\}$ satisfies the $(\pi^1,\pi^2,\dots,\pi^n)$-event aggregation property and $T_u=(r_1,r_2]$ and $|r_2-r_1|$ is sufficiently large, then

$$\sum_{j=1}^{n}\left\{\frac{\#(P_j,T_u)}{\pi^j}+1\right\}\ge \#(G_1,T_u).$$

Proof: Assume that a run is made in each of the P_j, $1\le j\le n$ and that the corresponding index sequences are $\alpha_{pj}\in A_{pj}$; the corresponding run in G_1 has index sequence $\alpha_{g1}\in A_{g1}$. Let $\alpha_{g1}(k')$ and $\alpha_{g1}(k'+1)$ be two elements on some $\alpha_{g1}\in A_{g1}$ such that $|\alpha_{g1}(k'+1)-\alpha_{g1}(k')|$ is an upper bound on the time between sequential events (that actually occurred) in G_1 for all possible runs in the P_j. Let $T_u=(r_1,r_2]$ such that $|r_2-r_1|>|\alpha_{g1}(k'+1)-\alpha_{g1}(k')|$. Assume that $\ell>0$ events occur in G_1 in T_u where $\ell=\ell_1+\ell_2+\cdots+\ell_n$ and ℓ_j is the number of events in G_1 that occurred due to a communication from P_j (if simultaneous events occur in P_j and P_k, $j\ne k$, causing an event occurrence in G_1, then the event occurrence can be attributed to either P_j or P_k and counted only once in ℓ). For each ℓ_j, at least $(\ell_j-1)\pi^j$ events have occurred in P_j, $1\le j\le n$. Therefore $\#(P_j,T_u)\ge(\ell_j-1)\pi^j$ so that

$$\sum_{j=1}^{n}\left\{\frac{\#(P_j,T_u)}{\pi^j}+1\right\}\ge \sum_{j=1}^{n}\ell_j=\ell=\#(G_1,T_u). \tag{8}$$

Clearly if any P_j deadlocks the relationship still holds. ∎

Corollary 1: If $\{P_j=(X^j,U^j,Y^j,\delta^j,\lambda^j,X_{0j}), \phi'_j: 1\le j\le n\}$ satisfies the $(\pi^1,\pi^2,\dots,\pi^n)$-event aggregation property and $T_u=(r_1,r_2]$ with $r_1=0$, then for all $r_2>0$,

$$\sum_{j=1}^{n}\left\{\frac{\#(P_j,T_u)}{\pi^j}\right\}\ge \#(G_1,T_u) \tag{9}$$

Corollary 2: For the multi-level HDES in Figure 3.2, if the (π^j)-event aggregation property holds for each successive level and $T_u=(r_1,r_2]$ where $|r_2-r_1|$ is sufficiently large, then for j, $1<j\le m$,

$$\frac{\#(G_j,T_u)}{\pi^j}+1\ge \#(G_{j+1},T_u). \tag{10}$$

A result similar to (10) holds for the multi-level HDES if T_u is chosen as in Corollary 2. The ϕ_j, ψ_j, and τ_j can be viewed as maps that cause event aggregation; consequently, Theorem 2 and Corollaries 1 and 2 provide a relationship between event aggregation and event rates for one class of HDES: *As events are aggregated to the higher levels in the hierarchy, fewer events occur.* If there is a high measure of aggregation at level j (large π^j) then there will be a lot fewer events occurring at level j+1. This illustrates that there is an inverse relationship between event aggregation and event rate between two levels of a HDES. In general, hierarchical systems researchers have observed a similar inverse relationship between "time scale density" ("time granularity") and "model abstractness" [2,13]. The above results provide the first mathematical validation of these researcher's intuition about relationships between event aggregation and event rates for a class of HDES. Next we provide an example to illustrate these results.

Example 3: *(Manufacturing System)* Here we consider a manufacturing system which consists of a machine that can process parts of two types, one at a time. The machine outputs each type part into a particular output bin and the machine can be idle. Let $X=\{MI, M_1, M_2, OUT_1, OUT_2\}$ be the set of states where MI means "machine idle", M_i means the machine is busy processing part type i, and OUT_i means that the machine outputs part type i. Let $U=\{u_1,u_2\}$ where u_i means input part type i into the machine. Let $Y=\{y_b, y_{1d}, y_{2d}\}$ where y_b indicates that the machine is busy processing a part of either type, and y_{id} indicates that the machine is done processing a part of type i. The transition function δ and the output function λ for the manufacturing system are specified via the bottom of Figure 3.3. We let $X_0=\{MI\}$ and consider $P_1=(X,U,Y,\delta,\lambda,X_0)$ to be our plant. We let $C_{p1}=E_{p1}$ and $\phi_1(ee)=e$ for all $ee\in E_{p1}^*$.

There is a higher level mechanism which forces the alternate processing of one part of type 1 and then two parts of type 2. This device is pictured in the top of Figure 3.3 and will be referred to as G_1. We have $G_1=(X_g,U_g,Y_g,\delta_g,\lambda_g,X_{0g})$ and $X_g=\{x_1,x_2,x_3\}$, $U_g=Y-\{y_b\}$, $Y_g=U$, and $X_{0g}=\{x_1\}$. Also, initially the input to the plant is u_1. The dotted arrows in Figure 3.3 are used to represent the timing structure and $\tau_1:C_{p1}\rightarrow\{0,1\}$ specifies the input trigger for G_1. For this example, we define $\phi_1(e(OUT_i,MI))=1$ for i=1, 2 and $\phi_1(\cdot)=0$ for all other cases. Notice that G_1 completely ignores output y_b as it is unimportant in coordinating the alternation of processing. (Hence, the inputs and outputs are not connected in a conventional manner where in G_1, $\delta_g(u,x)$ must be defined for all u.)

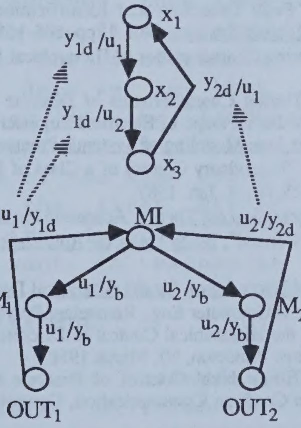


Figure 3.3 Model of a Manufacturing System

If the manufacturing system operates asynchronously (synchronously) then the coordination mechanism will operate asynchronously (with a general synchronous interpretation of time). Via Theorem 1, $\#(P_1,T_u)\geq\#(G_1,T_u)$ for all T_u. In particular we see that since event strings ending with (MI,M_i) and

(M_i,OUT_i) for $i=1,2$ are masked, a greater number of events will occur in the plant P_1 (lower level system) than in the controller G_1 (higher level system). Hence, for this simple manufacturing system the event rate at the higher level is lower no matter what the interpretation of time in P_1 is. Also, notice that if we choose $B_1=\{1,2\}$, $X_{i1}=\{M_i,OUT_i\}$ for $i\in B$, then $E_{a1}=\{(OUT_1,MI),(OUT_2,MI)\}$, and if we use the input trigger $\phi'_1:E_{a1}^*\rightarrow\{0,1\}$, a restriction of ϕ_1, and initial state as defined above, the conditions of Theorem 2 are satisfied with $\pi^1=2$ so $\#(P_1,T_u)/2 + 1\geq\#(G_1,T_u)$ for the proper T_u and all τ_1. Hence, via Theorem 2 we find an inverse relationship between event aggregation and event rate for this simple manufacturing system. ∎

Some of our other research focuses on the *design* of the timing structure to ensure that there is a decrease (by a some constant factor) in the event rate between any two levels of an HDES. A reduction in event rate is often desirable so that the processors implementing the higher level controls are permitted adequate time before they must attend to the lower level systems (e.g., take actions based on the occurrence of an event string).

4.0 REFERENCES

[1] Albus J.S., Barbera A.J., Nagel R.N., "Theory and Practice of Hierarchical Control", Proc. of the 23rd IEEE COMPCON, pp. 19-39, 1981.

[2] Antsaklis P.J., Passino K.M., Wang S.J., "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues", J. of Intelligent and Robotic Systems, Vol. 1, pp. 315-342, 1989.

[3] Gershwin S.B., "Hierarchical Flow Control: A Framework for Scheduling and Planning Discrete Events in Manufacturing Systems", Proc. of the IEEE, Vol. 77, No. 1, pp. 195-209, 1989.

[4] Knight J.F., Passino K.M., "Decidability for a Temporal Logic Used in Discrete Event System Analysis", To appear in the Int. Journal of Control.

[5] Mesarovic M., Macko D., Takahara Y., Theory of Hierarchical Multilevel Systems, Academic Press, NY, 1970.

[6] Passino K.M., Analysis and Synthesis of Discrete Event Regulator Systems, Ph.D. Dissertation, Dept. of Electrical Eng., University of Notre Dame, Notre Dame, IN, April 1989.

[7] Passino K.M., Antsaklis P.J., "Fault Detection and Identification in an Intelligent Restructurable Controller", J. of Intelligent and Robotic Systems, Vol. 1, pp. 145-161, 1988.

[8] Passino K.M., Antsaklis P.J., "Timing Characteristics of Hierarchical Discrete Event Systems", Submitted for Journal Publication, 1990.

[9] Passino K.M., Antsaklis P.J., "Timing Characteristics of Discrete Event Systems", Control Systems Technical Note #68, Univ. of Notre Dame, Dept. of Electrical Engineering, June 1989.

[10] Peterson J.L., Petri Net Theory and the Modeling of Systems, Prentice Hall, NJ, 1981.

[11] Ramadge P.J., Wonham W.M., "Supervisory Control of a Class of Discrete Event Processes", SIAM J. Control and Optimization, Vol. 25, No. 1, Jan. 1987.

[12] Sain M.K., Introduction to Algebraic System Theory, Academic Press, NY, 1981.

[13] Saridis G.N., "Intelligent Robot Control", IEEE Trans. on Automatic Control, Vol. AC-28, pp. 547-556, 1983.

[14] Valavanis K.P., A Mathematical Formulation for the Analytical Design of Intelligent Machines, Ph.D. Dissertation, Dept. of Electrical and Computer Eng., Rensselaer Polytechnic Inst., Troy, NY, Nov. 1986.

[15] Zhong H., Wonham W.M., "On the Hierarchical Control of Discrete-Event Systems", Proc. of the 1988 Conf. on Inf. Sciences and Systems, Princeton, NJ, March 1988.

[16] Zhong H., Wonham W.M., "Hierarchical Control of Discrete Event Systems: Computation and Examples", Proc. of the Allerton Conf. on Communication, Control, and Computing, Univ. of Illinois, Urbana, Sept. 1989.

[17] Zhong H., Wonham W.M., "On the Consistency of Hierarchical Supervision in Discrete-Event Systems", IEEE Trans. on Automatic Control, Vol. 35, No. 10, pp. 1125-1134, 1990.

A Qualitative Approach to Comparing Performance of Discrete Event Systems

Feng Lin
Department of Electrical and Computer Engineering
Wayne State University, Detroit, MI 48202

In this paper, we summarize a qualitative approach to comparing performance of discrete event systems proposed in [1-3].

Since most discrete event systems are complex systems one usually undertakes a qualitative analysis before performing a quantitative analysis. A theory of supervisory control has been developed over the past ten years to address qualitative issues in discrete event systems. It dose not, however, address quantitative issues directly. One problem of interest is how qualitative features of discrete event systems affect their quantitative features. Our aim is to investigate this problem. In other words, we would like to compare performance of discrete event systems by looking at their qualitative features.

Qualitative features of a discrete event system are usually described by the language that it generates. While its performance is described in terms of occurrence rates of its events. Our main result shows that if one system generates a larger language than another, then under a fairness condition, events occur faster in the first system than in the second. A result that relates the performance of a system to its qualitative features. Here we say that a language is "fair" if given two strings in the language with the number of occurrence of each event in the second no larger than the number of occurrence of the same event in the first, an event that may occur after the first string (i.e. the resulting string remains in the language) is also permitted to occur after the second. We want to emphasize that our result is independent of event lifetime distributions, which makes it interesting and significant.

This result has been extended to comparing queueing systems with structural differences. The performance of a queueing system depends on two things: (i) its structure, and (ii) its stochastic features. The structure of a queueing system is determined by the pattern of interconnection of various components of the system such as the way the servers are connected as well as things such as the buffer capacity of the system. Its stochastic features, on the other hand, are specified by things such as interarrival time and service time distributions. Because of the presence of both structural and stochastic features that are to be considered, such comparisons are nontrivial. It would thus be clearly of great interest if one could make such comparisons independent of the stochastic features. Our result in discrete event systems will help us in making such comparisons.

We define two classes of languages that we call "non-preemptive" and "order-independent". We say that a language is non-preemptive if when two different events are possible after a given string in the language, the occurrence of one cannot prevent the subsequent occurrence of the other, so that the occurrence of no event may preempt the occurrence of another event. A language is order-independent if given a string in the language, the order of occurrence of events in the string is irrelevant in determining whether a given event may follow that string. It can be easily verified that these two conditions hold in many queueing systems.

The two conditions, non-preemptiveness and order-independence together imply the fairness condition. Therefore, if both conditions are satisfied, a queueing system that generates larger language will perform better, in the sense that in this system events occur faster. This result has been applied to several queueing systems.

The first system that we consider is a closed cyclic queueing system with two single server stations. We can show that (1) the optimal number of customers in the system is always half the buffer capacity; (2) the best way to allocate buffer space is to split it into halves. The second system is a station with multiple servers. We can show that a single queue for all servers is better than multiple queues, one at each server. The third system is an open queueing network. We can show that the greater buffer capacity, the better the system performs.

[1] F. Lin, "That supervisor is best which supervises least", Preprint.

[2] F. Lin, "Stochastic comparison of queueing systems with structural differences: theory", Preprint.

[3] F. Lin, "Stochastic comparison of queueing systems with structural differences: application", Preprint.

Stability Analysis of Discrete Event Systems

Kevin M. Passino*,
Dept. of Electrical Eng.
The Ohio State Univ.
2015 Neil Ave.
Columbus, OH 43210

Anthony N. Michel*, and Panos J. Antsaklis**
Dept. of Electrical Eng.
Univ. of Notre Dame,
Notre Dame, IN 46556

Abstract

Discrete event systems (DES) are dynamical systems which evolve in time by the occurrence of events at possibly irregular time intervals. "Logical" DES are a class of discrete time DES with equations of motion that are most often non-linear and discontinuous with respect to event occurrences. Recently, there has been much interest in studying the stability properties of logical DES and several definitions for stability, and methods for stability analysis have been proposed. Here we introduce a logical DES model and define *stability in the sense of Lyapunov* for logical DES. Then we show that a more conventional analysis of stability which employs appropriate Lyapunov functions can be used for logical DES. This standard approach has the advantage of not requiring high computational complexity (as some of the others) but the difficulty lies in specifying the Lyapunov functions. The approach is illustrated on a manufacturing system that processes batches of N different types of parts according to a priority scheme and a load balancing problem in computer networks.

1.0 Introduction

Discrete event systems (DES) are dynamical systems which evolve in time by the occurrence of events at possibly irregular time intervals. Some examples include flexible manufacturing systems, computer networks, logic circuits, and traffic systems. "Logical" DES are a class of discrete time DES with equations of motion that are most often non-linear and discontinuous in the occurrence of the events. Recently, there has been much interest in studying the stability properties of logical DES and several definitions for stability, and methods for stability analysis have been proposed. Here we introduce a logical DES model and define *stability in the sense of Lyapunov* for logical DES. Then we show that the metric space formulation in [18] can be adapted so that a conventional analysis of stability which employs appropriate Lyapunov functions can be used for logical DES. An important advantage of the Lyapunov approach is that it does not require high computational complexity (as some of the other new approaches) but the difficulty lies in specifying the Lyapunov function. The approach is illustrated on a manufacturing system that processes batches of N different types of parts according to a priority scheme and a "load balancing problem" in computer networks. The full version of this paper, which includes the full proofs and other examples is given in [14].

The foundations for the study of stability properties of logical DES lie in the areas of general stability theory (the approach used herein) and theoretical Computer Science (recent DES-theoretic research). The two (related) main areas in theoretical Computer Science that form the foundation for logical DES-theoretic stability studies are temporal logic and automata. Intuitively speaking, in a temporal logic or automata-theoretic framework a system is considered in some sense *stable* if (i) for some set of initial states the system's state is

---

* The work of K.M. Passino and A.N. Michel was supported in part by the National Science Foundation under Grant ECS88-02924. Please address all correspondence to K.M. Passino.

** The work of P.J. Antsaklis was supported in part by the Jet Propulsion Laboratory.

guaranteed to enter a given set and stay there forever, or (ii) for some set of initial states the system's state is guaranteed to visit a given set of states infinitely often. In temporal logic, stability characteristics of logical DES have been studied in [4,16,5,6,10,13]. The automata theoretic work in Computer Science has also been adapted for the study of stability of DES in [12,11,3]. The construction of stabilizing controllers has also been studied in a Petri net framework in [7]. Certain general formulations for the study of stability are relevant to the study of stability properties of logical DES (e.g., see [17]). For an introduction to general stability theory and an overview of such research see [9]. Finally, in other DES studies, there have recently been significant advances in the study of stability properties of manufacturing systems in [15,8].

In Section 2 we introduce a logical DES model and in Section 3 we define stability in the sense of Lyapunov for DES and give necessary and sufficient conditions for stability of invariant sets of DES in a metric space. The applications are given in Section 4 and Conclusions in Section 5.

2.0 A Discrete Event System Model

We will consider stability properties of discrete event systems that can be accurately modelled with

$$G=(\mathcal{X},\mathcal{E},f_e,g,E_v) \tag{1}$$

where $\mathcal{X}$ is the set of states, $\mathcal{E}$ is the set of *events*,

$$f_e:\mathcal{X}\rightarrow\mathcal{X} \tag{2}$$

for $e\in\mathcal{E}$ are operators,

$$g:\mathcal{X}\rightarrow\mathbb{P}(\mathcal{E})-\{\emptyset\} \tag{3}$$

is the *enable function*, and $E_v\subset\mathcal{E}^{\mathbb{N}}$ is the set of *valid event trajectories*. Here, for an arbitrary set Z, $\mathbb{P}(Z)$ denotes the power set of Z. We only require that $f_e(x)$ be defined when $e\in g(x)$. We associate "time" indices with the states and events so that $x_k\in\mathcal{X}$ represents the state at time $k\in\mathbb{N}$ and $e_k\in\mathcal{E}$ represents an *enabled* event at time $k\in\mathbb{N}$ if $e_k\in g(x_k)$. If at state $x_k\in\mathcal{X}$, event $e_k\in\mathcal{E}$ *occurs* at time $k\in\mathbb{N}$ (randomly, not necessarily according to any particular statistics) then the next state x_{k+1} is given by application of the operator f_{e_k}, i.e., $x_{k+1}=f_{e_k}(x_k)$. Events can only occur if they lie on valid event trajectories as we now discuss.

Any sequence $\{x_k\}\in\mathcal{X}^{\mathbb{N}}$ such that for all k, $x_{k+1}=f_{e_k}(x_k)$ where $e_k\in g(x_k)$, is a *state trajectory*. The set of all *event trajectories* denoted with **E** is composed of those sequences $\{e_k\}\in\mathcal{E}^{\mathbb{N}}$ such that there exists a state trajectory $\{x_k\}\in\mathcal{X}^{\mathbb{N}}$ where for all k, $e_k\in g(x_k)$. Hence, to each event trajectory, which specifies the order of the application of the operators f_e, there corresponds a unique state trajectory (but, in general, not vice versa). The set of *valid event trajectories* $E_v\subset E$ represents the event trajectories that are physically possible in G. Hence, even if $x_k\in\mathcal{X}$ and $e_k\in g(x_k)$ it is not the case that e_k can occur unless it lies on a valid event trajectory that ends at x_{k+1}, where $x_{k+1}=f_{e_k}(x_k)$. In Section 4 we shall see that the use of E_v

can facilitate the modelling of many DES and provide flexibility in the study of stability properties. Let $\mathbf{E}_v(x_0)\subset\mathbf{E}_v$ denote the set of all possible valid event trajectories that begin from state $x_0\in\mathcal{X}$. Below, we shall also utilize a special set of *allowed event trajectories* denoted with $\mathbf{E}_a$, where $\mathbf{E}_a\subset\mathbf{E}_v$, and allowed event trajectories that begin at state $x_0\in\mathcal{X}$ denoted by $\mathbf{E}_a(x_0)$.

Let E_k, for fixed $k\in\mathbb{N}$, denote an event sequence of k events that have occurred (by definition $E_0=\emptyset$). If $E_k=e_0,e_1,\cdots,e_{k-1}$ we let $E_kE\in\mathbf{E}_v(x_0)$ denote the *concatenation* of E_k and (the infinite sequence) $E=e_k,e_{k+1},\cdots$, i.e. $E_kE=e_0,e_1,\cdots,e_{k-1},e_k,e_{k+1},\cdots$. The function $X(x_0,E_k,k)$ will be used to denote the state reached from $x_0\in\mathcal{X}$ by application of event sequence E_k such that $E_kE\in\mathbf{E}_v(x_0)$. (By definition, $X(x_0,\emptyset,0)=x_0$ for all $x_0\in\mathcal{X}$.) For fixed x_0 and E_k, $X(x_0,E_k,k)$ shall be called a *motion.* We assume that for all $x_0\in\mathcal{X}$, if $E_kE_{k'}E\in\mathbf{E}_v(x_0)$

$$X(X(x_0,E_k,k),E_{k'},k')=X(x_0,E_{k+k'},k+k') \qquad (4)$$

for all $k,k'\in\mathbb{N}$, such that $k'\geq k$ where $E_{k+k'}=E_kE_{k'}$. This is the standard semi-group property for dynamical systems.

3.0 Necessary and Sufficient Conditions for the Stability of Invariant Sets of DESs in a Metric Space

The following adapts the formulation developed in [18] to the study of stability properties of systems represented by the logical DES model introduced above. Let $\rho:\mathcal{X}\times\mathcal{X}\rightarrow\mathbb{R}$ denote a *metric* on $\mathcal{X}$, and $\{\mathcal{X};\rho\}$ a *metric space.* Let $\mathcal{X}_z\subset\mathcal{X}$ and $\rho(x,\mathcal{X}_z)=\inf\{\rho(x,x'):x'\in\mathcal{X}_z\}$ denote the *distance* from point x to the set $\mathcal{X}_z$. By a *functional* we shall mean a mapping from an arbitrary set to $\mathbb{R}$.

Definition 1: The *r-neighborhood* of an arbitrary set $\mathcal{X}_z\subset\mathcal{X}$ is denoted by the set $S(\mathcal{X}_z;r)=\{x\in\mathcal{X}:0<\rho(x,\mathcal{X}_z)<r\}$ where $r>0$.

Definition 2: The set $\mathcal{X}_m\subset\mathcal{X}$ is called *invariant* with respect to (w.r.t) G if from $x_0\in\mathcal{X}_m$ it follows that $X(x_0,E_k,k)\in\mathcal{X}_m$ for all E_k such that $E_kE\in\mathbf{E}_v(x_0)$ and $k\in\mathbb{N}$.

Definition 3: A closed invariant set $\mathcal{X}_m\subset\mathcal{X}$ of G is called *stable in the sense of Lyapunov w.r.t.* $\boldsymbol{E_a}$ if for any $\varepsilon>0$ it is possible to find a quantity $\delta>0$ such that when $\rho(x_0,\mathcal{X}_m)<\delta$ we have $\rho(X(x_0,E_k,k),\mathcal{X}_m)<\varepsilon$ for all E_k such that $E_kE\in\mathbf{E}_a(x_0)$ and $k\in\mathbb{N}$. If furthermore $\rho(X(x_0,E_k,k),\mathcal{X}_m)\rightarrow 0$ for all E_k such that $E_kE\in\mathbf{E}_a(x_0)$ as $k\rightarrow\infty$, then the closed invariant set $\mathcal{X}_m$ of G is called *asymptotically stable w.r.t.* $\boldsymbol{E_a}$.

Definition 4: A closed invariant set $\mathcal{X}_m\subset\mathcal{X}$ of G is called *unstable in the sense of Lyapunov w.r.t.* $\boldsymbol{E_a}$ if it is not stable in the sense of Lyapunov w.r.t $\mathbf{E}_a$.

Definition 5: If the closed invariant set $\mathcal{X}_m \subset \mathcal{X}$ of G is asymptotically stable in the sense of Lyapunov w.r.t. $\mathbf{E}_a$, then the set $\mathcal{X}_a$ of all states $x_0 \in \mathcal{X}$ and $x_0 \notin \mathcal{X}_m$ having the property $\rho(X(x_0,E_k,k),\mathcal{X}_m) \to 0$ for all E_k such that $E_kE \in \mathbf{E}_a(x_0)$ as $k \to \infty$ is called the *region of asymptotic stability of* $\mathcal{X}_m$ *w.r.t.* $\mathbf{E}_a$.

Definition 6: The closed invariant set $\mathcal{X}_m \subset \mathcal{X}$ of G with region of asymptotic stability $\mathcal{X}_a$ is called *asymptotically stable in the large w.r.t.* $\mathbf{E}_a$ if $\mathcal{X}_a = \mathcal{X}$.

Remark 1: The above definitions provide a conventional characterization of stability for logical DES. Some more recent studies of various types of stability for logical DES are surveyed in the Introduction.

The following Theorem provides necessary and sufficient conditions for stability of the DES defined in (1). The proofs for these results are contained in [14].

Theorem 1: In order for a closed invariant set $\mathcal{X}_m \subset \mathcal{X}$ of G to be stable in the sense of Lyapunov w.r.t $\mathbf{E}_a$ it is necessary and sufficient that in a sufficiently small neighborhood $S(\mathcal{X}_m;r)$ of the set $\mathcal{X}_m$ there exists a specified functional V with the following properties:

(i) For sufficiently small $c_1 > 0$, it is possible to find a $c_2 > 0$ such that $V(x) > c_2$ for $x \in S(\mathcal{X}_m;r)$ and $\rho(x,\mathcal{X}_m) > c_1$.

(ii) For any $c_4 > 0$ as small as desired, it is possible to find a $c_3 > 0$ so small that when $\rho(x,\mathcal{X}_m) < c_3$ for $x \in S(\mathcal{X}_m;r)$ we have $V(x) \leq c_4$.

(iii) $V(X(x_0,E_k,k))$ is a non-increasing function for $k \in \mathbb{N}$, for $x_0 \in S(\mathcal{X}_m;r)$, for all $k \in \mathbb{N}$, as long as $X(x_0,E_k,k) \in S(\mathcal{X}_m;r)$ for all E_k such that $E_kE \in \mathbf{E}_a(x_0)$.

Corollary 1: If the closed invariant set $\mathcal{X}_m \subset \mathcal{X}$ of G is stable in the sense of Lyapunov w.r.t $\mathbf{E}_a$ then it is stable in the sense of Lyapunov w.r.t all $\mathbf{E}_a'$ such that $\mathbf{E}_a' \subset \mathbf{E}_a$.

Theorem 2: In order for a closed invariant set $\mathcal{X}_m \subset \mathcal{X}$ of G to be asymptotically stable in the sense of Lyapunov w.r.t. $\mathbf{E}_a$ it is necessary and sufficient that in a sufficiently small neighborhood $S(\mathcal{X}_m;r)$, of the set $\mathcal{X}_m$ there exists a specified functional V having properties (i), (ii), and (iii) of Theorem 1 and furthermore $V(X(x_0,E_k,k)) \to 0$ as $k \to \infty$ for all E_k such that $E_kE \in \mathbf{E}_a(x_0)$ and for all $k \in \mathbb{N}$ as long as $X(x_0,E_k,k) \in S(\mathcal{X}_m;r)$.

Corollary 2: If the closed invariant set $\mathcal{X}_m \subset \mathcal{X}$ of G is asymptotically stable w.r.t $\mathbf{E}_a$ then it is asymptotically stable w.r.t all $\mathbf{E}_a'$ such that $\mathbf{E}_a' \subset \mathbf{E}_a$.

4.0 Discrete Event System Applications

4.1 Manufacturing System

The first example that we shall consider is the manufacturing system shown in Figure 1 that processes batches of N different types of parts according to a priority scheme. There are N *producers* P_i, where $1 \leq i \leq N$, of parts of different types. The producers P_i place batches of their parts in their respective *buffers* B_i, where $1 \leq i \leq N$. These buffers B_i have *safe capacity limits* of b_i where $b_i > 0$, $1 \leq i \leq N$. Let x_i, $1 \leq i \leq N$, denote the number of parts in buffer B_i. Let x_i for $N+1 \leq i \leq 2N$ denote the number of P_{i-N} type parts in the machine. The machine can safely process less than or equal to M (where $M > 0$) parts of any type, at any time. As the machine finishes processing batches of P_i type parts they are placed in their respective *output bins* (P_i-bins). The producers P_i can only place batches of parts in their buffers B_i if $x_i < b_i$. Also, there is a priority scheme whereby batches of P_i type parts are only allowed to enter the machine if $x_j = 0$ for all j such that $j < i \leq N$, i.e. only if there are no parts in any buffers to the left of the B_i buffer. Next, we specify the DES model G for the manufacturing system.

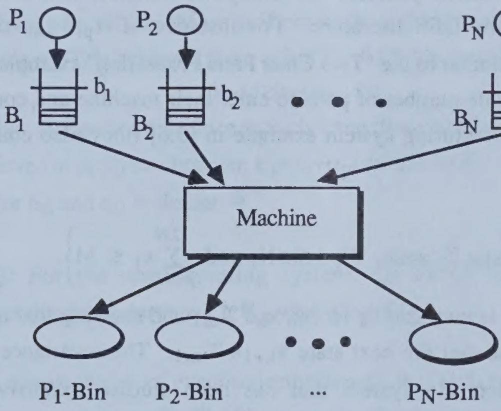


Figure 1. Manufacturing System with Priority Batch Processing

Let $\mathcal{X} = N^{2N}$ and $x_k \in \mathcal{X}$, where $x_k = [x_1\ x_2 \cdots x_N\ x_{N+1}\ x_{N+2} \cdots x_{2N}]^t$ (t denotes transpose) denote the state at time k. Let the set of events $\mathcal{E}$ be composed of events e_{pi} for $1 \leq i \leq N$ (representing the case where producer P_i places a batch of α_{pi} parts in buffer B_i), events e_{ai} for $1 \leq i \leq N$ (representing the case where a batch of α_{ai} P_i parts, from buffer B_i, arrive at the machine), and events e_{di} for $N+1 \leq i \leq 2N$ (representing the case where a batch of α_{di} P_i parts depart from the machine and are placed in their respective output bins). According to the above specifications the enable function g and event operators f_e for $e \in g(x_k)$ are defined as follows:

(i) If $x_i < b_i$ for any i, $1 \leq i \leq N$, then $e_{pi} \in g(x_k)$ and

$$f_{e_{pi}}(x_k) = [x_1\ x_2 \cdots x_i + \alpha_{pi} \cdots x_N\ x_{N+1}\ x_{N+2} \cdots x_{2N}]^t,$$

where $\alpha_{pi} \in \mathbb{N}-\{0\}$, $\alpha_{pi} \leq |x_i - b_i|$.

(ii) If $\sum_{j=N+1}^{2N} x_j < M$, and for some i, $1 \leq i \leq N$, $x_i > 0$, and $x_\ell = 0$ for all ℓ, $\ell < i \leq N$, then $e_{ai} \in g(x_k)$ and

$$f_{e_{ai}}(x_k) = [x_1\ x_2 \cdots x_i - \alpha_{ai} \cdots x_N\ x_{N+1}\ x_{N+2} \cdots x_{N+i} + \alpha_{ai} \cdots x_{2N}]^t,$$

where $\alpha_{ai} \in \mathbb{N}-\{0\}$, $\alpha_{ai} \leq x_i$, and

$$\alpha_{ai} \leq \left| \sum_{j=N+1}^{2N} x_j - M \right|.$$

(iii) If $x_i > 0$ for any i, $N+1 \leq i \leq 2N$, then $e_{di} \in g(x_k)$ and

$$f_{e_{di}}(x_k) = [x_1\ x_2 \cdots x_N\ x_{N+1}\ x_{N+2} \cdots x_{N+i} - \alpha_{di} \cdots x_{2N}]^t$$

where $\alpha_{di} \in \mathbb{N}-\{0\}$ and $\alpha_{di} \leq x_{N+i}$.

We let $\mathbf{E}_v = \mathbf{E}$, i.e. the set of all event trajectories is defined by g and f_e for $e \in g(x_k)$. The system operates in a standard asynchronous fashion.

This manufacturing system is a generalization of computer systems often used in the study of a simple "mutual exclusion problem" in Computer Science, and similar to several applications studied in the DES literature. For instance, if $\alpha_{pi} = \alpha_{ai} = \alpha_{di} = 1$ then our manufacturing system is similar to the "Two Class Parts Processing" example in [16] (except they allow an arbitrary finite number of parts to enter their machine and consider only two producers), and the manufacturing system example in [5,6] (they also consider only two producers).

Let

$$\mathcal{X}_{m1} = \left\{ x_k \in \mathcal{X} : x_i \leq b_i\ \forall i,\ 1 \leq i \leq N, \text{ and } \sum_{j=N+1}^{2N} x_j \leq M \right\}. \tag{5}$$

It is easy to see that $\mathcal{X}_{m1}$ is invariant by letting $x_k \in \mathcal{X}_{m1}$ and showing that no matter which event occurs it is the case that the next state $x_{k+1} \in \mathcal{X}_{m1}$. The invariance of $\mathcal{X}_{m1}$ is the property of the manufacturing system that has been studied extensively in similar manufacturing system examples [16,5,6]. Also, if M=1, N=2, $\alpha_{pi} = \alpha_{ai} = \alpha_{di} = 1$, and the priority scheme is removed, then the proof of the invariance of $\mathcal{X}_{m1}$ is equivalent to proving the mutual exclusion property often studied in Computer Science mentioned above.

Here, we provide a new study of the *stability properties* of the above manufacturing system. Intuitively this will, for instance, show that under certain conditions if the manufacturing system starts in an unsafe operating mode (too many parts in a buffer or in the machine, or both) it will eventually return to a safe operating condition.

Proposition 1: For the manufacturing system, the closed invariant set $\mathcal{X}_{m1}$ is stable in the sense of Lyapunov w.r.t $\mathbf{E}_a$, where $\mathbf{E}_a = \mathbf{E}_v$, for any neighborhood of $\mathcal{X}_{m1}$ (i.e., any r>0).

Proof: Let $\mathbf{x}_k=[x_1 \cdots x_{2N}]^t$, $\mathbf{x}_{k+1}=[x'_1 \cdots x'_{2N}]^t$, $\bar{\mathbf{x}}=[\bar{x}_1 \cdots \bar{x}_{2N}]^t$, and $\bar{\mathbf{x}}'=[\bar{x}'_1 \cdots \bar{x}'_{2N}]^t$. Choose

$$\rho(\mathbf{x}_k,\mathcal{X}_{m1})=\inf\left\{\sum_{j=1}^{2N}|x_j-\bar{x}_j| : \bar{\mathbf{x}}\in\mathcal{X}_{m1}\right\} \tag{6}$$

and $V_1(\mathbf{x}_k)=\rho(\mathbf{x}_k,\mathcal{X}_{m1})$. Conditions (i) and (ii) of Theorem 1 follow directly from the choice of $V_1(\mathbf{x}_k)$. For condition (iii) we show that $V_1(\mathbf{x}_k)\geq V_1(\mathbf{x}_{k+1})$ for all $\mathbf{x}_k\notin\mathcal{X}_{m1}$ no matter what event $e\in g(\mathbf{x}_k)$ occurs causing $\mathbf{x}_{k+1}=f_e(\mathbf{x}_k)$, so long as it lies on an event trajectory in $\mathbf{E}_a$. For $\mathbf{x}_k\notin\mathcal{X}_{m1}$ if e_{pi} occurs for some i, $1\leq i\leq N$, then we need to show that

$$\inf\left\{\sum_{j=1}^{2N}|x_j-\bar{x}_j| : \bar{\mathbf{x}}\in\mathcal{X}_{m1}\right\} \geq \inf\left\{\sum_{\substack{j=1\\ j\neq i}}^{2N}|x_j-\bar{x}'_j|+|x_i+\alpha_{pi}-\bar{x}'_i| : \bar{\mathbf{x}}'\in\mathcal{X}_{m1}\right\}. \tag{7}$$

It suffices to show that for all $\bar{\mathbf{x}}\in\mathcal{X}_{m1}$ at which the inf is achieved on the left of (7) there exists $\bar{\mathbf{x}}'\in\mathcal{X}_{m1}$ such that

$$\sum_{j=1}^{2N}|x_j-\bar{x}_j| \geq \sum_{\substack{j=1\\ j\neq i}}^{2N}|x_j-\bar{x}'_j|+|x_i+\alpha_{pi}-\bar{x}'_i| \tag{8}$$

If we choose $\bar{x}'_\ell=\bar{x}_\ell$ for all $\ell\neq i$ then it suffices to show that for all $\bar{x}_i$, $0\leq\bar{x}_i\leq b_i$, at which the inf on the left side of (7) is achieved there exists $\bar{x}'_i$, $0\leq\bar{x}'_i\leq b_i$, such that

$$|x_i-\bar{x}_i|\geq|x_i+\alpha_{pi}-\bar{x}'_i| \tag{9}$$

where $\alpha_{pi}\leq|x_i-b_i|$. But notice that since $0\leq x_i<b_i$ even though $0\leq\bar{x}_i\leq b_i$ the inf on the left of (7) will not be achieved at $\bar{x}_i=b_i$ so choosing $\bar{x}'_i=\bar{x}_i+\alpha_{pi}$ results in $\bar{\mathbf{x}}'\in\mathcal{X}_{m1}$ and the satisfaction of (9). The case for e_{ai} and e_{di} is similar. ■

Proposition 2: For the manufacturing system, the closed invariant set $\mathcal{X}_{m1}$ is not asymptotically stable in the large w.r.t. $\mathbf{E}_a$, where $\mathbf{E}_a=\mathbf{E}_v$.

Let $\mathbf{E}'_a\subset\mathbf{E}_a$ denote the set of event trajectories such that each event $e\in\mathcal{E}$ occurs infinitely often on each event trajectory $E\in\mathbf{E}'_a$. If we assume for the manufacturing system that only events which lie on event trajectories in $\mathbf{E}'_a$ occur, then it is always the case that eventually each type of event (e_{pi}, e_{ai}, $1\leq i\leq N$, and e_{di}, $N+1\leq i\leq 2N$) will occur.

Proposition 3: For the manufacturing system, the closed invariant set $\mathcal{X}_{m1}$ is asymptotically stable in the large w.r.t $\mathbf{E}'_a$.

The use of the set $\mathbf{E}'_a$ for the manufacturing system imposes what is called a "fairness" constraint in Computer Science (in our example we require that each producer P_i get fair use of the machine). Such constraints are used in the study of temporal logic, the mutual exclusion problem in Computer Science, and in [17] when the author studies conditions under which the Lyapunov function can be constructed mechanically for a class of logical DES.

4.2 Computer Network Load Balancing Problem

Consider a network of computers described by an undirected graph (C,A) where $C=\{1,2,\ldots,N\}$ represents a set of computers that are numbered with $i\in C$, and $A\subset C\times C$ is the set of connections between the computers. We require that if $i\in C$ then there exists $(i,j)\in A$ (or $(j,i)\in A$) for some $j\in C$ (i.e., every computer is connected to the network). Also, if $(i,j)\in A$ then $(j,i)\in A$. Each computer has a buffer which holds tasks (load) each of which can be executed by any computer in the network. Let the load of computer $i\in C$ be given by x_i; hence, $x_i\geq 0$. Each connection in the network $(i,j)\in A$ allows for computer i to pass a portion of its load to computer j and vice-versa. It also allows computer i and computer j to sense the size of the load of one another (any two computers i and j such that $(i,j)\notin A$ may not pass portions of their loads or sense the values of each others loads).

We assume that initially the distribution of the load across the computers is uneven and seek to prove properties relating to the system achieving a more even distribution of tasks so that the computers in the network are more fully utilized. For convenience we assume that the computers will not begin working on any of the tasks or receive any more to process until the load has been balanced (This assumption is easily lifted and our analysis still applies as we discuss below in Remark 2). Next we specify the model G for the computer network load balancing problem.

Let $\mathcal{X}=\mathbb{N}^N$ denote the set of states and $\mathbf{x}_k=[x_1\ x_2\ \cdots\ x_N]^t$ and $\mathbf{x}_{k+1}=[x'_1\ x'_2\ \cdots\ x'_N]^t$ denote the state at time k and k+1 respectively. Let $\mathcal{E}=\{e_\alpha^{ij}: (i,j)\in A, \alpha\in\mathbb{N}-\{0\}\}\cup\{e_0\}$ be the set of events for G where e_α^{ij} represents the case where α amount of load of computer i is transferred to computer j and e_0 represents an null event (i.e., that no load is transfered). Let $M\in\mathbb{N}-\{0\}$ be the amount of load imbalance tolerated between any two computers i and j where $(i,j)\in A$. Next we specify g and f_e for $e\in g(\mathbf{x}_k)$:

(i) If for any $(i,j)\in A$, $|x_i-x_j|>M$ then

- (a) if $x_i>x_j$, then $e_\alpha^{ij}\in g(\mathbf{x}_k)$ and $f_e(\mathbf{x}_k)=\mathbf{x}_{k+1}$ where $e=e_\alpha^{ij}$, $x'_i:=x_i-\alpha$, $x'_j:=x_j+\alpha$, $x'_k:=x_k$ for all $k\neq i,j$, and $0<\alpha\leq(1/2)|x_i-x_j|$ for $\alpha\in\mathbb{N}$.
- (b) if $x_j>x_i$, then $e_\alpha^{ji}\in g(\mathbf{x}_k)$ and $f_e(\mathbf{x}_k)=\mathbf{x}_{k+1}$ where $e=e_\alpha^{ji}$, $x'_j:=x_j-\alpha$, $x'_i:=x_i+\alpha$, $x'_k:=x_k$ for all $k\neq i,j$, and $0<\alpha\leq(1/2)|x_i-x_j|$ for $\alpha\in\mathbb{N}$.

(ii) If $|x_i-x_j|\leq M$ for all $(i,j)\in A$ then $e_0\in g(\mathbf{x}_k)$ and $f_{e_0}(\mathbf{x}_k)=\mathbf{x}_k$.

Let $\mathbf{E}_v=\mathbf{E}$ and $\mathcal{X}_{m2}=\{\mathbf{x}_k\in\mathcal{X}: |x_i-x_j|\leq M \text{ for all } (i,j)\in A\}$ which is clearly invariant.

This load balancing problem is similar to the one in [1] except they require the load of the computers to be represented by a continuous variable, seek a perfect balance of tasks, allow a computer to simultaneously pass load to several neighboring computers, and allow for the possibility that a computer's information about the load of adjacent computers is outdated. They also require that the system is "partially asynchronous" so that they can achieve load balancing when the computers only have possibly outdated information about neighboring loads. Various forms of the load balancing problem have also been studied in the DES literature [2] and extensively studied in the Computer Science literature (See [1,2] and the references therein).

The following Proposition provides a new characterization and analysis of the Lyapunov stability of the computer network load balancing problem described above.

Proposition 4: For the computer network load balancing problem, the closed invariant set $\mathcal{X}_{m2}$ is asymptotically stable in the large w.r.t. $\mathbf{E}_a$, where $\mathbf{E}_a = \mathbf{E}_v$.

Proof: Let $\bar{\mathbf{x}} = [\bar{x}_1 \cdots \bar{x}_N]^t$, $\bar{\mathbf{x}}' = [\bar{x}'_1 \cdots \bar{x}'_N]^t$, and choose

$$\rho(\mathbf{x}_k, \mathcal{X}_{m2}) = \inf\left\{ \max\left\{ |x_1 - \bar{x}_1|, \cdots, |x_N - \bar{x}_N| \right\} : \bar{\mathbf{x}} \in \mathcal{X}_{m2} \right\}$$

and $V_2(\mathbf{x}_k) = \rho(\mathbf{x}_k, \mathcal{X}_{m2})$ so that conditions (i) and (ii) of Theorem 1 are satisfied. For condition (iii) of Theorem 1 it is shown in [14] that for all $\mathbf{x}_k \notin \mathcal{X}_{m2}$ and all $e_\alpha^{ij} \in g(\mathbf{x}_k)$ when e_α^{ij} occurs $V_2(\mathbf{x}_k) \geq V_2(\mathbf{x}_{k+1})$ and that the invariant set is asymptotically stable in the large w.r.t. $\mathbf{E}_a$. ■

Remark 2: If tasks enter the computer network or get processed by one of the computers $i \in C$ we let a new initial state $\mathbf{x}_0$ reflect the increased or decreased load and the above stability analysis shows that the load will still eventually balance. (This characteristic was also discussed in [1].) In fact, if the total amount of load is finite then it will take a finite amount of time for the load to become balanced.

5.0 Conclusions

It has been shown that it is possible to define and study Lyapunov stability of a wide class of logical DES by adapting the metric space formulation in [18]. Hence, logical DES, which have recently received much attention in the literature are amenable to conventional stability analysis via the choosing of appropriate Lyapunov functions. Other notions of stability and more recent stability analysis techniques based on methods from theoretical Computer Science (surveyed in the Introduction) are often prohibitive due to problems with computational complexity. Here, we completely avoid problems with computational complexity but instead rely on the specification of Lyapunov functions that satisfy certain properties. We have shown that it is not difficult to specify such Lyapunov functions for two types of DES applications: a manufacturing system that processes batches of N different types of parts according to a priority scheme and a load balancing problem in computer networks. Our characterization of stability of DES in a traditional stability-theoretic framework will, in the future, allow researchers to use the vast body of concepts from the field of Lyapunov stability theory to study properties of DES.

6.0 References

[1] Bertsekas D.P., Tsitsiklis J.N., Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Engelwood Cliffs, NJ, 1989.

[2] Boel R.K., van Schuppen J.H., "Distributed Routing for Load Balancing", Proceedings of the IEEE, Vol. 77, No. 1, pp. 210-221, 1989.

[3] Brave Y., Heymann M., "On Stabilization of Discrete Event Processes", Proceeding of the 28th Conf. on Dec. and Control, pp. 2737-2742, Tampa, FL, Dec. 1989.

[4] Fusaoka A., Seki H., and Takahashi K., "A Description and Reasoning of Plant Controllers in Temporal Logic", Proc. of the 8th Int. Joint Conf. on Artificial Intelligence, pp. 405-408, August 1983.

[5] Knight J.F., Passino K.M., "Decidability for Temporal Logic in Control Theory", Proc. of the Allerton Conf. on Communication, Control, and Computing, Univ. of Illinois, Urbana-Champaign, pp. 335-344, Oct. 1987.

[6] Knight J.F., Passino K.M., "Decidability for a Temporal Logic Used in Discrete Event System Analysis", To appear in the Int. Journal of Control 1990.

[7] Krogh B.H., "Controlled Petri Nets and Maximally Permissive Feedback Logic", Proc. of the 25th Allerton Conf., Univ. of Illinois at Urbana-Champaign, Sept. 1987.

[8] Kumar P.R., Seidman T.I., "Dynamic Instabilities and Stabilization Methods in Distributed Real-Time Scheduling of Manufacturing Systems", IEEE Trans. on Automatic Control, Vol. 35, No. 3, March 1990.

[9] Michel A.N., Miller R.K., Qualitative Analysis of Large Scale Dynamical Systems, Academic Press, NY, 1977.

[10] Ostroff J. S., Real-Time Computer Control of Discrete Systems Modelled by Extended State Machines: A Temporal Logic Approach, PhD Dissertation, Report No. 8618, Dept. of Elect. Eng., Univ. of Toronto, Jan. 1987.

[11] Ozveren C.M., Analysis and Control of Discrete Event Dynamic Systems: A State Space Approach, Ph.D. Dissertation, MIT, LIDS Report LIDS-TH-1907, Aug. 1989.

[12] Ozveren C.M., Willsky A.S., Antsaklis P.J., "Stability and Stabilizability of Discrete Event Dynamic Systems", MIT, LIDS Report LIDS-P-1853, Feb. 1989 (To appear in the Journal of the ACM).

[13] Passino K.M., Antsaklis P.J., "Branching Time Temporal Logic for Discrete Event System Analysis", Proc. of the 26th Allerton Conf. on Communication, Control, and Computing, pp. 1160-1169, Univ. of Illinois at Urbana-Champaign, Sept. 1988.

[14] Passino K.M., Michel A.N., Antsaklis P.J., "Lyapunov Stability of a Class of Discrete Event Systems", Submitted for Journal Publication, 1990.

[15] Perkins J.R., Kumar P.R., "Stable, Distributed, Real-Time Scheduling of Flexible Manufacturing/Assembly/Disassembly Systems", IEEE Trans. on Automatic Control, Vol. 34, No. 2, Feb. 1989.

[16] Thistle J. G., Wonham W. M., "Control Problems in a Temporal Logic Framework", International Journal Control, Vol. 44, No. 4, pp. 943-976, 1986.

[17] Tsitsiklis J.N., "On the Stability of Asychronous Iterative Processes", Mathematical Systems Theory, Vol. 20, pp. 137-153, 1987.

[18] Zubov V.I., Methods of A.M. Lyapunov and Their Application, P. Noordhoff Ltd., The Netherlands, 1964.

ON THE THEORY OF POLYNOMIAL MATRIX INTERPOLATION AND ITS ROLE IN SYSTEMS AND CONTROL

Panos J. Antsaklis and Zhiqiang Gao
Dept. of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

Introduction

In this paper, new theoretical results on polynomial and rational matrix interpolation are briefly outlined and their application to certain Systems and Control problems is summarized.

The use of interpolation type constraints in system and control theory is first discussed and a number of examples are presented. With this motivation, an appropriate formulation for polynomial matrix interpolation is introduced via its basic theorem, Theorem 1. It is shown that this generalization, of polynomial interpolation to polynomial matrix, is most general and it includes all other such interpolation schemes which have appeared in the literature. The basic theorem of rational matrix interpolation is also presented. A specialization of these results to involve characteristic values and vectors of polynomial matrices is shown; note that the eigenvalue, eigenvector problem of a real matrix can be seen as a special case. Equations involving polynomial and rational matrices are of great interest in systems and control. It is shown how to solve these equations using interpolation theory. Note that certain of these results have appeared in [1-4] and used in [5-6] and elsewhere. Here, for the first time, the overall approach is outlined and it is connected to interpolation conditions used by many in the system and control literature. This is in addition to new contributions in the polynomial matrix interpolation theory. It is hoped that this paper will provide some insight and will demonstrate the application of the polynomial matrix interpolation theory to the theory and practice of systems and control.

Motivation: Interpolation type constraints in Systems and Control theory

Many control system constraints and properties that are expressed in terms of conditions on a polynomial or rational matrix R(s), can be written in an easier to handle form in terms of $R(s_j)$, where $R(s_j)$ is R(s) evaluated at certain (complex) values $s = s_j$ $j = 1, \ell$. We shall call such conditions in terms of $R(s_j)$, interpolation (type) conditions on R(s). This is because in order to understand the exact implications of these constraints on the structure and properties of R(s), one needs to use results from polynomial interpolation theory. Note that although for the scalar polynomial case, interpolation is an old and very well studied problem, only recently polynomial matrix interpolation appears to have been addressed in any systematic way [1-3].

Next, a number of examples from Systems and Control theory where polynomial and polynomial matrix interpolation constraints are used, are outlined. This list is not complete, by far.

It is known that all the uncontrollable eigenvalues of $\dot{x} = Ax + Bu$ are given by the roots of the determinant of a greatest left divisor of the polynomial matrices $sI - A$ and B. An alternative, and perhaps easier to handle, form of this result is that s_j is an uncontrollable eigenvalue if and only if there exists real row vector v_j such that $v_j[s_jI - A, B] = 0$ (PBH controllability test). This is a more restrictive version of the previous result which involves left divisors, since it is not clear how to handle multiple eigenvalues when it is desirable to determine all uncontrollable eigenvalues. The results presented here can readily provide the solution to this problem.

Consider an example from control design: Given the plant $P(s)$ and a controller $G(s)$, the closed loop transfer function $T(s)$ is given by $T(s) = (I + G(s)P(s))^{-1}G(s)P(s)$ assuming a unity feedback configuration. The design problem is now to find a controller $G(s)$ if $T(s)$ and $P(s)$ are given. The method of Ragazzini (see description in [7] pp. 216-218) handles the scalar case. In particular, for stability, all unstable poles of $P(s)$ must be zeros of $1 - T(s)$; and all unstable zeros of $P(s)$ must be zeros of $T(s)$ (note that $T(s)$ is stable). Furthermore for causality, $T(s)$ must have as its zeros at infinity all the zeros at infinity of $P(s)$. Clearly, the condition $1 - T(s_j) = 0$ where s_j $j = 1, \ell$ are the unstable poles of $P(s)$ is an alternative form of the first constraint on $T(s)$; the case of multiple poles at s_j can be handled by differentiating. Similarly for the remaining constraints. The corresponding multivariable stability constraints were introduced in [8] and they are: $T(s) = N(s)X(s)$ and $X(s)$, $[I - X(s)N(s)]D^{-1}(s)$ stable where $P(s) = N(s)D^{-1}(s)$ a coprime polynomial factorization. These can also be expressed as interpolation constraints, however care should be taken when $P(s)$ has multiple poles. Note that for a square polynomial matrix $D(s)$ to have a zero of determinant at some value s_j, $D(s_j)a_j = 0$ where a_j is a nonzero (complex) vector. While in the polynomial case $d'(s_j) = 0$ is a necessary and sufficient condition for $d(s)$ to have a second zero at s_j, a derivative of $D(s)$ is not necessary in the matrix case.

The state feedback eigenvalue assignment problem has a rather natural formulation in terms of interpolation type constraints; similarly the output feedback problem[4-6].

More recently, stability constraints in the H^∞ formulation of the optimal control problem have been expressed in terms of interpolation type constraints[9-11]. It is rather interesting that [9-10] discuss a "directional" approach which is in the same spirit of the approach we take here (and in [1-7]).

The above are just a few of the many examples of the strong presence of interpolation type conditions in the systems and control literature; this is because they represent a convenient way to handle certain types of constraints. However, a closer look reveals that the relationships between conditions on $R(s_j)$ and properties of the matrix $R(s)$ are not clear at all and this needs to be explained. Only in this way one can take full advantage of the method and develop new approaches to handle control problems. Our research on matrix interpolation and its applications attempts to address this need.

Results from Polynomial Matrix Interpolation Theory

In the following some of the basic results of the polynomial and rational matrix interpolation theory are outlined. Note that this method, in addition to offering a systematic approach to solve certain system control problems, perhaps more importantly, it is a completely new theory developed from the bottom up and it opens a new field of study.

Basic Theorem of Polynomial Matrix Interpolation

The basic theorem of polynomial interpolation can be stated as follows: Given ℓ distinct complex points s_j $j = 1, \ell$ and ℓ complex values b_j there exists a unique $(\ell-1)$th degree polynomial $q(s)$ for which

$$q(s_j) = b_j \qquad j = 1, \ell \tag{1}$$

A generalization of this result to polynomial matrices is as follows:

Let $S(s) := \text{blk diag}\,[\,1, s, \ldots, s^{d_i}]'$ where d_i $i = 1, m$ are non-negative integers; let $a_j \neq 0$ and b_j denote $(m\times 1)$ and $(p\times 1)$ complex vectors respectively and s_j complex scalars.

Theorem 1: Given (s_j, a_j, b_j) $j = 1, \ell$ and nonnegative integers d_i with $\ell = \sum d_i + m$ such that the square $(\sum d_i + m)\times\ell$ matrix

$$S_\ell = [S(s_1)\, a_1, \ldots, S(s_\ell) a_\ell] \tag{2}$$

has full rank, there exists a unique $(p\times m)$ polynomial matrix $Q(s)$, with ith column degree equal to d_i, $i = 1, m$ for which

$$Q(s_j) a_j = b_j \qquad j = 1, \ell \tag{3}$$

It should be noted that when $p = m = 1$ and $d_1 = \ell-1$ this theorem reduces to the polynomial interpolation theorem. To see this, note that in this case the nonzero scalars a_j $j = 1, \ell$, can be taken to be equal to 1, in which case S_ℓ is exactly the well known Vandermonde matrix which is nonsingular if and only if s_j are distinct.

Proof: Since the column degrees of $Q(s)$ are d_i, $Q(s)$ can be written as

$$Q(s) = QS(s) \tag{4}$$

where $Q(p\mathrm{x}(\Sigma d_i + m))$ contains the coefficients of the polynomial entries. Substituting in (3), Q must satisfy

$$QS_{\ell} = B_{\ell} \tag{5}$$

where $B_{\ell} = [b_1, ..., b_{\ell}]$. Since S_{ℓ} is nonsingular, Q and therefore Q(s) are uniquely determined. □

Example 1: Let Q(s) be 1x2 (=pxm) and let $\ell = 3$ interpolation points be specified: $\{(s_j, a_j, b_j)\ j = 1, 2, 3\} = \{ (-1,[1, 0]', 0), (0,[-1, 1]', 0), (1, [0, 1]', 1)\}$. Q(s) is uniquely specified when d_1 and d_2 are chosen so that $\ell(=3) = \Sigma d_i + m = (d_1 + d_2) + 2$ or $d_1 + d_2 = 1$ and S_3 has full rank. Clearly the resulting Q(s) depends on the particular choice for the column degrees d_i and different combinations of d_i will result to different matrices Q(s):

(i) Let $d_1 = 1$, and $d_2 = 0$. Then $S(s) = \text{blk diag}[[1,s]',1]$ and (5) becomes:

$$Q\,S_3 = Q\begin{bmatrix} 1 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} = [0, 0, 1]$$

from which $Q = [1, 1, 1]$ and $Q(s) = QS(s) = [s+1, 1]$.

(ii) Let $d_1 = 0$, $d_2 = 1$. Then $S(s) = \text{blk diag}[1, [1, s]']$ and (2.5) gives $Q = [0, 0, 1]$ from which $Q(s) = [0, s]$, clearly different from (i) above. □

Note that alternative polynomial bases, other than $[1, s, s^2, ...]'$, which might offer computational advantages in determining Q(s) from interpolation equations (5) can of course be used (and they have); for example, Tschebyscheff polynomials.

The relation between (4) and an alternative, also commonly used representation of Q(s), is now shown, namely:

$$Q(s) = Q_d\, S_d(s) \tag{6}$$

where $S_d(s) := [I, ..., Is^d]'$ an $m(d+1)\mathrm{x}m$ matrix with $d = \max(d_i)$ $i = 1, m$. Notice that $S(s) = KS_d(s)$ where K $((\Sigma d_i + m) \times m(d + 1))$ describes the appropriate interchanges of rows in S_d (s) needed to extract S(s). Representation (6) can be used in matrix interpolation as the following corollary shows:

Corollary 1: Given (s_j, a_j, b_j) $j = 1, \ell$ and nonnegative integer d with $\ell = m(d+1)$ such that the $m(d+1)\mathrm{x}\ell$ matrix $S_{d\ell} = [S_d(s_1)\,a_1, ..., S_d(s_{\ell})a_{\ell}]$ has full rank, there exists a unique (pxm) polynomial matrix Q(s) with highest polynomial entry degree d which satisfies (3).

Example 2 Let $Q(s)$ be 1x2 (= pxm), $d = 1$ and let the $\ell = m(d+1) = 4$ interpolation points (s_j, a_j, b_j) be as follows: let the first 3 be the same as in Example 1 and the fourth be (2, [0, 1]', 1). The equation $Q_d S_{d\ell} = B_\ell$ now becomes

$$Q_d S_{d\ell} = Q_d \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix} = [0, 0, 1, 1] = B_\ell$$

from which $Q_d = [1, 1, 1, 0]$ and $Q(s) = Q_d S_d(s) = [s+1, 1]$ as in Example 2.1 (i). If the fourth interpolation point is taken to be equal to (2, [0, 1]', 2) then $B_\ell = [0, 0, 1, 2]$ while $S_{d\ell}$ remains the same. Then $Q_d = [0, 0, 0, 1]$ and $Q(s) = Q_d S_d(s) = [0, s]$ as in Example 1(ii). ☐

Interpolation constraints of the form

$$Q(z_k) = Q_k \quad k = 1, q \tag{7}$$

have also appeared in the literature. These conditions are but a special case of (3). In fact for each k, (7) represents m special conditions of the form (3). To see this, consider (3) and blocks of m interpolation points where $s_i = z_1$ $i = 1, m$ with $a_i = e_i$, $s_{m+i} = z_2$ $i = 1, m$ with $a_{m+i} = e_i$ and so on, where the entries of e_i are zero except the ith entry which is 1; then Q_1 of (7) above is $Q_1 = [b_1, ..., b_m]$, $Q_2 = [b_{m+1}, ..., b_{2m}]$ and so on. In this case s_j are not distinct but the value is repeated m times. A simple comparison of the constraints (7) to the polynomial constraints (1) seems to suggest that this is an attempt to directly generalize the scalar results to the matrix case. As in the polynomial case, z_k $k = 1, q$ therefore should perhaps be distinct for $Q(s)$ to be uniquely determined. Indeed this is the case as the following corollary shows:

Corollary 2: Given (z_k, Q_k) $k = 1, q$ with $q = d + 1$, and Q_k (pxm), such that the m(d+1)xmq matrix $S_{dk} := [S_d(z_1), ..., S_d(z_k)]$ has full rank, there exists a unique (pxm) polynomial matrix $Q(s)$ with highest polynomial entry degree d which satisfies (7).

Example 3 Let $Q(s)$ be 1x2 (=pxm), $d = 1$ and let the $q = d+1 = 2$ interpolation points be $\{z_k, R_k \; k = 1, 2\} = \{(0, [1, 1]), (1, [2, 1])\}$. In view of $Q(s) = Q_d S(s)$, $Q_d[S_d(z_1), S_d(z_2)] = [R_1, R_2]$ or

$$Q_d \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [1, 1, 2, 1]$$

from which $Q_d = [1, 1, 1, 0]$ and $Q(s) = Q_d S_d(s) = [s+1, 1]$ as in Examples 1 and 2. ☐

Characteristic Values and Vectors

If a complex number z satisfies $q(z) = 0$, where $q(s)$ is a polynomial, then z is a root of $q(s)$. In view of this, let us apply the basic matrix interpolation Theorem 2.1 with $b_j = 0$; that is the pxm matrix $Q(s)$ must satisfy

$$Q(s_j)\, a_j = 0 \qquad j = 1, \ell \tag{8}$$

Definition 1: A complex scalar s_j is a characteristic value of $Q(s)$ if it is a zero of an invariant polynomial of $Q(s)$. The mx1 complex vector a_j which satisfies $Q(s_j)a_j = 0$ is the corresponding characteristic vector of $Q(s)$.

$Q(s)$ may have repeated characteristic values so s_j typically has an algebraic and a geometric multiplicity defined below for $Q(s)$ nonsingular; it is straightforward to extend the definitions to a general pxm $Q(s)$. In the case of a real matrix A, if some of the eigenvalues are repeated one may have to use generalized eigenvectors. Here we also define generalized characteristic vectors of $Q(s)$. In the results below, only characteristic vectors which satisfy relation (8), which does not contain derivatives of $Q(s)$, are considered for simplicity and clarity. The general results are not included here; as perhaps expected, they involve derivatives of $Q(s)$[1].

Let $Q(s)$ be an (mxm) nonsingular matrix. If s_j is a zero of $|Q(s)|$ repeated n_j times, define n_j to be the *algebraic multiplicity* of s_j; define also the *geometric multiplicity* of s_j as the quantity $(m - \text{rank}\, Q(s_j))$.

Theorem 2: There exist a complex scalar s_j and ℓ_j linearly independent (mx1) vectors a_{ij} with $i = 1, \ell_j$ which satisfy

$$Q(s_j)\, a_{ij} = 0 \tag{9}$$

if and only if s_j is a zero of $|Q(s)|$ with algebraic multiplicity $(=n_j) \geq \ell_j$ and geometric multiplicity $(=(m - \text{rank}Q(s_j)) \geq \ell_j$.

The complex values s_j and vectors a_{ij} are characteristic values and vectors of $Q(s)$. In the case when $\ell_j = 1$, the theorem simply states that s_j is a zero of $|Q(s)|$ if and only if $\text{rank}Q(s_j) < m$, an obvious and well known result. The conditions of Theorem 2 imply certain structure for the Smith form of $Q(s)$. In particular (9) implies that the Smith form of $Q(s)$ has factors $(s - s_j)$, raised to some power, in ℓ_j locations on the diagonal[1].

In the following, matrices $Q(s)$ with given characteristic values and vectors s_j and a_{ij} are characterized.

Theorem 3: Let $n = \deg|Q(s)|$. There exist σ distinct scalars s_j and $(m\times1)$ nonzero vectors a_{ij} $i = 1, \ell_j$ $j = 1, \sigma$ with $\sum_1^{\sigma} \ell_j = n$ and a_{ij} $i = 1, \ell_j$ linearly independent which satisfy (9) if and only if the zeros of $|Q(s)|$ have σ distinct values s_j $j = 1, \sigma$, each with algebraic multiplicity $n_j = \ell_j$ and geometric multiplicity $(m - \text{rank}\, Q(s_j)) = \ell_j$.

Note that the independence condition on the $m\times1$ vectors a_{ij} implies that $\ell_j \leq m$, that is no characteristic value is repeated more that m times in Theorem 3; derivatives must be used if s_j is repeated more than m times[1]. The following corollary of Theorem 3 formalizes the most familiar case:

Corollary 3: Let $n = \deg |Q(s)|$. There exist n distinct complex scalars s_j and $(m\times1)$ nonzero vectors a_j $j = 1, n$ which satisfy (8) if and only if the zeros of $|Q(s)|$ have n distinct values s_j.

Rational Matrix Interpolation :

The polynomial matrix interpolation Theorem 1 directly leads to rational matrix interpolation theorem:

Theorem 4: Assume that (s_j, c_j, b_j) $j = 1, \ell$ $c_j \neq 0$ with $\ell = \sum d_i + m$ and a polynomial matrix $D(s)$ with $|D(s_j)| \neq 0$ are given, such that the S_ℓ matrix in (2) with $a_j = [D(s_j)]^{-1} c_j$ has full rank. There exists a unique rational matrix $H(s)$ of the form $H(s) = N(s)D(s)^{-1}$, where the polynomial matrix $N(s)$ has column degrees $\deg_{ci}[N(s)] = d_i$, for which

$$H(s_j)\, c_j = b_j \qquad j = 1, \ell \tag{10}$$

Rational matrix interpolation results can be directly derived from corresponding polynomial matrix interpolation results. All results of polynomial matrix interpolation can therefore be extended to the rational matrix case. Notice that if $D(s)$, in Theorem 4, is taken to be I, Theorem 1 is obtained. One could also use the results of Corollary 1 and 2 to obtain alternative rational matrix interpolation results.

Solutions of Polynomial and Rational Matrix Equations

Consider the equation

$$M(s)L(s) = Q(s) \tag{11}$$

where $L(s)$ (txm) and $Q(s)$ (kxm) are given polynomial matrices. The polynomial matrix interpolation theory developed above will now be used to solve this equation and determine the polynomial matrix solutions $M(s)$ (kxt).

First consider the left hand side of equation (11). Let $M(s) := M_0 + ... + M_r s^r$ where r is a non-negative integer, and let $d_i := \deg_{ci}[L(s)]$ $i = 1, m$. If $\hat{Q}(s) := M(s)L(s)$, then $\deg_{ci}[\hat{Q}(s)] = d_i + r$. According to the basic polynomial matrix interpolation Theorem 2.1, the matrix $\hat{Q}(s)$ can be uniquely specified using $\sum (d_i+r) + m = \sum d_i + m(r+1)$ interpolation points. Therefore consider ℓ interpolation points (s_j, a_j, b_j) $j = 1, \ell$ where

$$\ell = \sum d_i + m(r+1) \tag{12}$$

and such that the theorem assumptions are satisfied; that is, if $S_r(s) :=$ blk diag$[1, s, ..., s^{d_i+r}]'$, the $(\sum d_i + m(r+1))$xℓ matrix $S_{r\ell} := [S_r(s_1)\, a_1, ..., S_r(s_\ell)a_\ell]$ has full rank. Note that for distinct s_j, $S_{r\ell}$ will have full column rank for almost any set of nonzero a_j (this can be formally shown). Now in view of Theorem 1 $\hat{Q}(s)$ which satisfies

$$\hat{Q}(s_j)a_j = b_j \tag{13}$$

is uniquely specified given these ℓ interpolation points (s_j, a_j, b_j). To solve (11), these interpolation points must be appropriately chosen so that the equation $\hat{Q}(s)$ $(= M(s)L(s)) = Q(s)$ is satisfied:

Write (11) as

$$ML_r(s) = Q(s) \tag{14}$$

where $M := [M_0, ..., M_r]$ (kxt(r+1)) and $L_r(s) := [L(s)', ..., s^rL(s)']'$ (t(r+1)xm). Let $s = s_j$ and postmultiply (14) by a_j $j = 1, \ell$. Define

$$b_j := Q(s_j)a_j \quad j = 1, \ell \tag{15}$$

and combine the equations to obtain

$$ML_{r\ell} = B_\ell \tag{16}$$

where $L_{r\ell} := [L_r(s_1)\, a_1, ..., L_r(s_\ell)a_\ell]$ (t(r+1)xℓ) and $B_\ell := [b_1, ..., b_\ell]$(kx$\ell$).

Theorem 5: Given $L(s)$, $Q(s)$ in (11), let $d_i := \deg_{ci}[L(s)]$ $i = 1, m$ and select r to satisfy

$$\deg_{ci}[Q(s)] \leq d_i + r \quad i = 1, m \tag{17}$$

Then a solution $M(s)$ of degree r exists if and only if a solution M of (16) does exist; furthermore, $M(s) = M[I, sI, ..., s^rI]'$.

It is not difficult to show that solving (16) is equivalent to solving

$$M(s_j)c_j = b_j \quad j = 1, \ell \tag{18}$$

where

$$c_j := L(s_j)a_j,\ b_j := Q(s_j)a_j \quad j = 1, \ell \tag{19}$$

In view now of Corollary 1, the matrices M(s) which satisfy (18) are obtained by solving

$$MS_\ell = B_\ell \tag{20}$$

where $S_\ell := [S(s_1)\ c_1, \ldots, S(s_\ell)c_\ell]$ $(t(r+1)\mathrm{x}\ell)$, with $S(s) := [I, sI, \ldots, s^rI]'$ $(t(r+1)\mathrm{x}t)$ and $B_\ell := [b_1, \ldots, b_\ell]$ $(k\mathrm{x}\ell)$; M(s) is then $M(s) = M[I, sI, \ldots, s^rI]'$ where M $(k\mathrm{x}t(r+1))$ satisfies (20). Solving (20) is an alternative to solving (16).

Theorem 5 shows that there is a one-to-one mapping between the solutions of degree r of the polynomial matrix equation (11) and the solutions of the linear system of equations (16) (or of (20)). In other words, using (16) (or (20)), we can characterize all solutions of degree r of (11). Note that the conditions (17) of the theorem are not restrictive as they are necessary conditions for a solution M(s) in (11) of degree r to exist; that is, all solutions of M(s)L(s) = Q(s) of any degree can be found using Theorem 5. Also note that no assumptions were made regarding the polynomial matrices in (11), that is Theorem 5 is valid for any matrices L(s), Q(s) of appropriate dimensions. Solving (16) (or (20)) is equivalent to solving (11) for solutions M(s) of degree ≤ r. When applying this approach, it is not necessary to determine in advance a lower bound for r; it suffices to use a large enough r. Theorem 5 provides the theoretical guarantee that in this way all solutions of (11) can be obtained. Searching for solutions is straightforward in view of the availability of computer software packages to solve linear system of equations. Even when an exact solution does not exist, it can be approximated using, for example, least squares.

Now let's consider the rational matrix equation:

$$M(s)L(s) = Q(s) \tag{21}$$

where L(s) (txm) and Q(s) (kxm) are given rational matrices. The polynomial matrix interpolation theory developed above will now be used to solve this equation and determine the rational matrix solutions M(s) (kxt). Let $M = D^{-1}(s)N(s)$, a polynomial fraction form of M(s) to be determined; then equation (21) can be written as:

$$[N(s) \quad D(s)]\begin{bmatrix} L(s) \\ -\ Q(s) \end{bmatrix} = 0 \tag{22}$$

Let $s = s_j$ and postmultiply by a_j $j = 1, \ell$ with a_j and ℓ chosen properly (see below). Let

$$c_j := [\ L(s_j)', ,\ -\ Q(s_j)']'\ a_j \quad j = 1, \ell \tag{23}$$

The problem now is to find a polynomial matrix [N(s) D(s)] which satisfies

$$[N(s_j) \quad D(s_j)]\, c_j = 0 \qquad j = 1, \ell \tag{24}$$

which is a polynomial interpolation problem that can be solved using the approaches proposed above. In fact (24) is of the form of (18). The question really is how to select a_j $j = 1, \ell$ and ℓ so that by solving (24) all solutions to (21) are obtained:

Note that instead of solving (22) one could equivalently solve [N(s) D(s)][Lp(s)' -Qp(s)']' = 0, where [Lp(s)' -Qp(s)']' is a polynomial numerator of [L(s)' -Q(s)']'; so the problem to be solved is now (11), a polynomial matrix equation, where L(s) = [Lp(s)' -Qp(s)']' and Q(s) = 0. Therefore, Theorem 5 does apply and all solutions [N(s) D(s)] of degree r can be determined by solving (16) or (18) which in this case is (24) above.

Note that restrictions can be easily imposed to guarantee that $D^{-1}(s)$ exists and that $M(s) = D^{-1}(s)N(s)$ is proper. The existence of solutions of (21) and their causality depends on the given rational matrices L(s) and Q(s) (see for example [12]). Our approach here will find a proper rational matrix of order r when such solution exists. Additional interpolation type constraints can be added so the solution satisfies additional specifications.

References

[1] P.J. Antsaklis, "Polynomial Matrix Characterization Using Characteristic Values and Vectors", Publ. No. 80/18, Dept. of Electr. Engr., Imperial College, London, July 1980.

[2] P.J. Antsaklis, "Polynomial Matrix Interpolation in Control Equations: The Diophatine Equation", Proc. of the 1983 Conf. on Info. Sciences and Systems, pp. 873-878, The Johns Hopkins University, March 1983.

[3] P.J. Antsaklis and R.L. Lopez, "Control Design Using Polynomial Matrix Interpolation", Proc. of the 23rd IEEE Conf. on Decision and Control, pp 621-622, Las Vegas, Nevada, Dec. 12-14, 1984.

[4] P.J. Antsaklis, Some New Matrix Methods Applied to Multivariable System Analysis and Design, Ph.D. Dissertation, Brown University, 1976.

[5] P.J. Antsaklis and W. A. Wolovich, "Arbitrary Pole Placement Using Linear Output Feedback Compensation", Int. J. Control, Vol. 25, pp. 915-925, 1977.

[6] R. Lopez, Multivariable Control Design via Polynomial Interpolation Methods, M.S.E.E. Thesis, Dept. of Electr. Engr., University of Notre Dame, May 1984.

[7] G.F. Franklin, J.D. Powell and M.L. Workman, Digital Control of Dynamic Systems, 2nd Edition, Addison-Wesley Publishing Company, Reading, MA, 1990.

[8] P.J. Antsaklis and M.K. Sain, "Unity Feedback Compensation of Unstable Plants", Proc. of the 20th IEEE Conf. on Decision and Control, pp. 305-308, San Diego, CA, Dec. 1981.

[9] H. Kimura, "Directional Interpolation Approach to H^∞ -Optimization and Robust Stabilization", IEEE Transactions on Automatic Control, Vol. AC-32, No. 12, pp.1085-1093, December 1987.

[10] U. Shaked, "The Structure of Inner Matrices that Satisfy Multiple Directional Interpolation Requirements", IEEE Transactions on Automatic Control, Vol. AC-34, No. 12, pp. 1293-1296, December 1989.

[11] B.-C. Chang and J. B. Pearson, Jr., "Optimal Disturbance Reduction in Linear Multivariable Systems", IEEE Transactions on Automatic Control, Vol. AC-29, No. 10, pp. 880-887, October 1984.

[12] Z. Gao and P.J. Antsaklis, "On Stable Solutions of the One and Two Sided Model Matching Problems", IEEE Transactions on Automatic Control, Vol. AC-34, No. 9, pp. 978-982, September 1989.

CANONICAL FORMS FOR LINEAR TIME-VARYING MULTIVARIABLE DISCRETE SYSTEMS

Baeil P. Park and Erik I. Verriest

School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT: This paper presents state-space canonical forms for *linear (generally) time-varying* (LTV) multivariable completely reachable discrete-time systems. The result generalizes the time-invariant and periodically time-varying cases, respectively.

I. INTRODUCTION:

Although some studies on LTV systems were already done, for examples, the algebraic description [1][2], the general expression of single variable difference equations [3] and others [4][5], this paper presents the state-space approach as an extension of canonical forms for *linear periodically time- varying* (LPTV) systems [6][7]. This extension is inspired by that the Fourier series technique for periodic signals is extended to the Fourier transform for nonperiodic signals by considering the limit as the period goes to infinity. The developed framework would be useful toward the studies of nonstationary stochastic signals modeling and processing, adaptive filtering and control, and nonlinear systems modeling by linearization.

Throughout the paper, LTV systems are defined on the parameter set $\{((A_k, B_k, C_k, D_k))\}$ of sequences of quadruples in the state-space form

$$x(k+1) = A_k x(k) + B_k u(k) \tag{1}$$
$$y(k) = C_k x(k) + D_k u(k) \tag{2}$$

where for all k

$$(A_k, B_k, C_k, D_k) \in R^{n\times n} \times R^{n\times m} \times R^{p\times n} \times R^{p\times m} \tag{3}$$

If two sequences $((A_k, B_k, C_k, D_k))$ and $((A'_k, B'_k, C'_k, D'_k))$ are similar i.e. there exists a sequence (T_k) of nonsingular matrices such that

$$A_k T_{k-1} = T_k A'_k \quad B_k = T_k B'_k \quad C_k T_{k-1} = C'_k \quad D_k = D'_k \tag{4}$$

then they are called to be in the time-varying similarity equivalence relation R^{LTV}, and the corresponding two LTV systems are clearly *input/output* (I/O) equivalent. The reachability matrix $R_r((A_k, B_k))$ at r of a sequence $((A_k, B_k, C_k, D_k))$ is defined for all $r \geq 0$

$$R_r((A_k, B_k)) = [B_r \;\; A_r B_{r-1} \;\; \dots \;\; A_r A_{r-1} \cdots A_1 B_0] \tag{5}$$

If the reachability matrix $R_r((A_k, B_k))$ is full rank for all $r \geq$ some integer μ, then the corresponding LTV system is called completely reachable, and denoted by $\sum_{n,m,p,\infty}^{cr}$. This reachability definition is general and thereby valid for the LTI and LPTV cases. The parameter set $\{((A_k, B_k, C_k, D_k)) : R_r((A_k, B_k))$ is full rank for all $r \geq$ some $\mu\}$ is especially denoted by Θ^{LTV}. Therefore, the completely reachable LTV systems $\{\sum_{n,m,p,\infty}^{cr}\}$ are defined on the parameter set Θ^{LTV}.

A canonical form is generally a map Γ associated with a given parameter set Θ, objective function $F : \Theta \rightarrow S$ and equivalence relation R: If F is invariant under R i.e. for all $\theta, \theta' \in \Theta$

$$\theta R \theta' \;\Rightarrow\; F(\theta) = F(\theta') \tag{6}$$

and if a map $\Gamma : \Theta \rightarrow \Theta$ satisfies the two conditions for all $\theta, \theta' \in \Theta$

$$\Gamma(\theta) \;\; R \;\; \theta \tag{7}$$

$$\theta \; R \; \theta' \quad \Leftrightarrow \quad \Gamma(\theta) = \Gamma(\theta') \tag{8}$$

then the map Γ is called a canonical form for $F : \Theta \rightarrow S$ under R. This paper pursues a canonical form $\Gamma^{LTV} : \Theta^{LTV} \rightarrow \Theta^{LTV}$ for $F^{LTV} : \Theta^{LTV} \rightarrow S$ under R^{LTV} such that

$$\Theta^{LTV} \equiv \{((A_k, B_k, C_k, D_k)) : R_r((A_k, B_k)) \text{ is full rank for all } r \geq \text{ some } \mu\} \tag{9}$$

$$F^{LTV} \equiv \text{the LTV systems } \{\textstyle\sum_{n,m,p,\infty}^{cr}\} \text{ I/O operation (1) and (2)} \tag{10}$$

$$R^{LTV} \equiv \text{the time-varying similarity-equivalence in (4)} \tag{11}$$

Note that F^{LTV} is invariant under R^{LTV}. The goal of this paper is now well defined to find a map Γ^{LTV} that satisfies the two conditions (7) and (8). In what follows, the mapping algorithm of such map Γ^{LTV} is explicitly shown as an extension from the LTI case to the LTV one through the LPTV result as mentioned above.

II. LTI CANONICAL FORMS : LTI systems are defined by (1) and (2) along with the time- invariance of the parameter in (3) and the similarity in (4)

$$(A_k, B_k, C_k, D_k) = (A, B, C, D) \tag{12}$$

$$T_k = T \tag{13}$$

Due to the Cayley-Hamiltion theorem, the reachability matrix $R(A, B)$ of a quadruple (A, B, C, D) can simply be redefined from the above general definition

$$R(A, B) = [B \;\; AB \;\; \dots \;\; A^{n-1}B] \tag{14}$$

It is now a well known fact [8][9] that a canonical form $\Gamma^{LTI} : (A, B, C, D) \mapsto (A', B', C', D')$ for such time- invariant (completely) reachable systems $\{\sum_{n,m,p,1}^{cr}\}$ is realized by a mapping rule with the three steps:

Step (i): Search n independent columns in the full rank reachability matrix $R(A,B)$ with an order (see Fig. 1-a or b).
Step (ii): Rearrange those n independent columns with the fixed order (see Fig. 1-a) to form a nonsingular matrix T.
Step (iii): Using the similarity transformation in (4)(12) and (13), find the canonical quadruple (A',B',C',D').

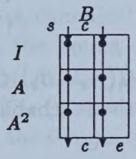


Fig. 1-a: the LTI Young diagram of the scheme I: $n=3, m=2$ case

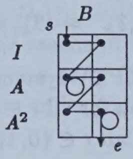


Fig. 1-b: the LTI Young diagram of the scheme II: $n=3, m=2$ case

Indeed, a map of this type of mapping rules satisfies the two conditions (7) and (8) to be a canonical form. For the details of the steps (i) and (ii), recall the Young diagram associated with a reachability matrix $R(A,B)$ as in Fig. 1-a or b, where each cell defined by A^k row and $[B]_i$ column represents the column $A^k[B]_i$ of the reachability matrix $R(A,B)$ in (14). The scanning scheme I shown in Fig. 1-a defines the fixed rearranging order for the step (ii). A scheme either I in Fig. 1-a or II in Fig. 1-b defines the search order for the step (i). If the search order is defined with the scheme II, the corresponding map is called the reachability canonical form $\Gamma^{LTI,2}$ with the scheme II. For an example, suppose that during the search with the scheme II the two cells $A[B]_1$ and $A^2[B]_2$ marked by ○'s in Fig. 1-b turn out to be dependent on the previous cells

$$\begin{aligned} A[B]_1 &= a_1[B]_1 + a_2[B]_2 \\ A^2[B]_2 &= a_3[B]_1 + a_4[B]_2 + a_5 A[B]_2 \end{aligned}$$

The transformation matrix T is formed with the Scheme I in Fig. 1-a

$$T = [[B]_1 \ [B]_2 \ A[B]_2]$$

Applying (4)(12) and (13) to the above three equations, the canonical pair (A',B') is

$$A' = \begin{bmatrix} a_1 & 0 & a_3 \\ a_2 & 0 & a_4 \\ 0 & 1 & a_5 \end{bmatrix} \qquad B' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{15}$$

where all fixed $0's$ and $1's$ are determined by the index $(1,2)$ that specifies the location of the dependent cells marked by ○'s on the diagram. Such index is called a structural invariant one associated with the canonical form $\Gamma^{LTI,2}$. An ordered set $(a_1, a_2, \ldots, a_5)$

of the real numbers in (15) is called a parametric invariant value. The other canonical pair (C', D') does not have any fixed entries.

III. LPTV CANONICAL FORMS : The equations (1) and (2) along with the periodicity of the parameter in (3) and the similarity in (4)

$$(A_k, B_k, C_k, D_k) = (A_{k+N}, B_{k+N}, C_{k+N}, D_{k+N}) \tag{16}$$

$$T_k = T_{k+N} \tag{17}$$

define the completely reachable LPTV systems $\{\sum^{cr}_{n,m,p,N}\}$ on $\Theta^{LPTV} = \{((A_k\ , B_k\ , C_k\ , D_k))_N : R_r((A_k, B_k))_N$ is full rank for all $r \in \{0, 1, \ldots, N-1\}\}$, where the reachability matrix $R_r((A_k, B_k))_N$ at r is for all $r \in \{0, 1, \ldots, N-1\}$

$$R_r((A_k, B_k))_N \equiv \left[\beta_r \ \alpha_r\beta_r \ \ldots \ \alpha_r^{n-1}\beta_r\right] \tag{18}$$

where

$$\alpha_r \equiv A_r \cdots A_0 A_{N-1} \cdots A_{r+1} \tag{19}$$

$$\beta_r \equiv [B_r \ A_rB_{r-1} \ A_rA_{r-1}B_{r-2} \quad \ldots \quad A_r \cdots A_0A_{N-1} \cdots A_{r+2}B_{r+1}\ ] \tag{20}$$

Note that the Cayley-Hamilton theorem is applicable to the system matrices (A_k). Now, let us consider a canonical form Γ^{LPTV} for $\{\sum^{cr}_{n,m,p,N}\}$. The mapping algorithm can be sketched as the followings: By the definition of the complete reachability, n independent columns can be found in each $R_r((A_k, B_k))_N$, and a nonsingular matrix T_r can be formed from those independent columns. Finally, using the similarity transformation in (4)(16) and (17), a new N-tuple $((A'_k, B'_k, C'_k, D'_k))_N$ of quadruples can be obtained. A map of such mapping algorithm satisfies the two conditions (7) and (8). However, it is necessary to design a mapping algorithm that gives a priori information of, for example, the fixed structure in the resulting N-tuple as in the time-invariant case (15). In other words, an algorithm should provides well defined invariants associated with the map. Recently, such 'canonical' algorithms are introduced (see [6][7] for the great details of this section) based on the equivalent quadruple (A_e, B_e, C_e, D_e) of a N-tuple $((A_k, B_k, C_k, D_k))_N$ of quadruples

$$A_e \equiv \begin{bmatrix} 0 & \cdots & 0 & A_0 \\ A_1 & & & 0 \\ & \ddots & & \vdots \\ & & A_{N-1} & 0 \end{bmatrix} \quad B_e \equiv \begin{bmatrix} B_0 & & & \\ & B_1 & & \\ & & \ddots & \\ & & & B_{N-1} \end{bmatrix}$$

$$C_e \equiv \begin{bmatrix} 0 & \cdots & 0 & C_0 \\ C_1 & & & 0 \\ & \ddots & & \vdots \\ & & C_{N-1} & 0 \end{bmatrix} \quad D_e \equiv \begin{bmatrix} D_0 & & & \\ & D_1 & & \\ & & \ddots & \\ & & & D_{N-1} \end{bmatrix} \tag{21}$$

where all unspecified block entries are zero. This setting is indeed very convenient. The reachability matrices $\{R_r((A_k, B_k))_N \text{ for all } r \in \{0, 1, \dots, N-1\}\}$ in (18) to (20) are well ordered in the equivalent reachability matrix $R(A_e, B_e)$

$$R(A_e, B_e)T_c = diag\ (R_0((A_k, B_k))_N \ \ R_1((A_k, B_k))_N \ \ \dots \ \ R_{N-1}((A_k, B_k))_N) \qquad (22)$$

where T_c is a column interchange (permutation) matrix.

A canonical form $\Gamma^{LPTV} : ((A_k, B_k, C_k, D_k))_N \mapsto ((A'_k, B'_k, C'_k, D'_k))_N$ is realized using the Young diagram associated with the reachability matrix $R(A_e, B_e)$ and thereby $\{R_r((A_k, B_k))_N\}$ in (18) to (20). See Fig. 2-a or b. The reachability matrix $R_0((A_k, B_k))_N$ (with the proper zero- augmentation as in (22)), for example, is represented by all the 0-numbered cells. More specifically,

$$A_e^k[B_e]_{li} \ \Leftrightarrow \ \overbrace{A_r A_{r-1} \ \cdots \ A_{r-k+1}}^{k}[B_l]_i \qquad (23)$$

where all subscripts stand in the modulo-N sense, and the number r is determined by

$$r \ = \ (k + l) \ mod \ N \qquad (24)$$

For an example, the cell $A_e^2[B_e]_{11}$ marked by ○ in Fig. 2-b corresponds (is equal) to the column $A_0 A_2[B_1]_1$ of $R_0((A_k, B_k))_N$ (with the zero- augmentation). A canonical algorithm is now given with the three steps:

Step (i): Search n independent columns in $R_r((A_k, B_k))_N$ for each r with an order (see Fig. 2-a or b).
Step (ii): Rearrange n independent columns with the fixed order (see Fig. 2-a) to form a nonsingular matrix T_r.
Step (iii): Apply the similatity transformation in (4)(16) and (17) to obtain the canonical N- tuple $((A'_k, B'_k, C'_k, D'_k))_N$ of quadruples.

As in the above LTI case, if the scheme *ii* in Fig. 2-b is used for the step (ii), the corresponding canonical form is called the complete-reachability canonical form $\Gamma^{LPTV,2}$ with the scheme *ii*. For the first example, suppose that during the search with the scheme *ii* the six cells $A_e[B_e]_{01}$, $A_e^2[B_e]_{02}$, $A_e[B_e]_{11}$, $A_e^2[B_e]_{12}$, $A_e[B_e]_{21}$ and $A_e^2[B_e]_{22}$, see (23) and (24), are found to be dependent on the previous cells

$$\begin{aligned}
A_0[B_2]_1 &= a_{0,1}[B_0]_1 + a_{0,2}[B_0]_2 \\
A_0 A_2[B_1]_2 &= a_{0,3}[B_0]_1 + a_{0,4}[B_0]_2 + a_{0,5}A_0[B_2]_2 \\
A_1[B_0]_1 &= a_{1,1}[B_1]_1 + a_{1,2}[B_1]_2 \\
A_1 A_0[B_2]_2 &= a_{1,3}[B_1]_1 + a_{1,4}[B_1]_2 + a_{1,5}A_1[B_0]_2 \\
A_2[B_1]_1 &= a_{2,1}[B_2]_1 + a_{2,2}[B_2]_2 \\
A_2 A_1[B_0]_2 &= a_{2,3}[B_2]_1 + a_{2,4}[B_2]_2 + a_{2,5}A_2[B_1]_2
\end{aligned}$$

The transformation matrices T_0, T_1 and T_2 are formed with the scheme i in Fig. 2-a

$$\begin{aligned} T_0 &= [[B_0]_1 \; [B_0]_2 \; A_0[B_2]_2] \\ T_1 &= [[B_1]_1 \; [B_1]_2 \; A_1[B_0]_2] \\ T_2 &= [[B_2]_1 \; [B_2]_2 \; A_2[B_1]_2] \end{aligned}$$

Applying (4)(16) and (17) to the above nine equations, the canonical N-tuple $((A'_k, B'_k))_N$ of pairs turns out to be for $k = 0, 1$ and 2

$$A'_k = \begin{bmatrix} a_{k,1} & 0 & a_{k,3} \\ a_{k,2} & 0 & a_{k,4} \\ 0 & 1 & a_{k,5} \end{bmatrix} \qquad B'_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{25}$$

where all $0's$ and $1's$ are completely determined by the location $(1,2,1,2,1,2)$ of the dependent cells on the diagram in Fig. 2-b. For the second example, if nN cells in $(1,3,2,0,0,3)$ marked by $\bigcirc$'s turn out to be dependent on the previous cells with the scheme ii, the canonical N- tuple $((A''_k, B''_k))_N$ is

$$A''_0 = \begin{bmatrix} a_{0,1} & 0 & a_{0,4} \\ a_{0,2} & 0 & a_{0,5} \\ a_{0,3} & 1 & a_{0,6} \end{bmatrix} \quad A''_1 = \begin{bmatrix} a_{1,1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad A''_2 = \begin{bmatrix} 1 & 0 & a_{2,1} \\ 0 & 0 & a_{2,2} \\ 0 & 1 & a_{2,3} \end{bmatrix}$$

$$B''_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad B''_1 = \begin{bmatrix} 1 & a_{1,2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad B''_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{26}$$

The other canonical N-tuples $((C'_k, D'_k))_N$ and $((C''_k, D''_k))_N$ do not have any fixed structures as in the LTI case.

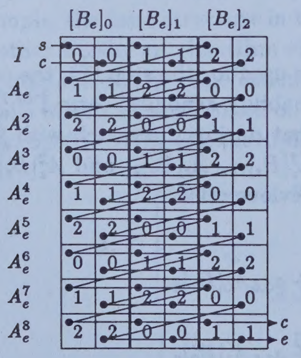


Fig. 2-a: the LPTV Young diagram of the scheme i: n=3, m=2 and N=3 case

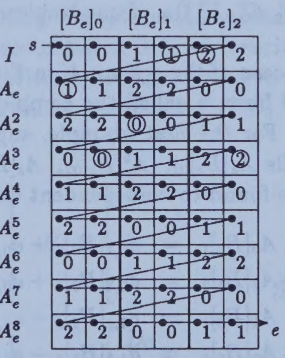


Fig. 2-b: the LPTV Young diagram of the scheme ii: n=3, m=2 and N=3 case

Remark 1: In fact, a LPTV system $\sum_{n,m,p,N}$ defined on $((A_k, B_k, C_k, D_k))_N$ is instantaneous-I/O and asymptotic-stability equivalent to the decomposition model $S^* \sum^e_{nN,mN,pN,1} S$. The systems S^* and S are periodic switches and inverse to each other, and $\sum^e_{nN,mN,pN,1}$ is the LTI system defined on (A_e, B_e, C_e, D_e) in the state-space form [6][7].

IV. LTV CANONICAL FORMS: Since the Cayley-Hamilton theorem cannot be applied to a sequence of the system matrices (A_k) , the dimension of reachability matrix $R_r((A_k, B_k))$ at each r of a sequence $((A_k, B_k, C_k, D_k))$ is not fixed as shown in (5). However, by the definition of complete reachability, n independent columns can be still found in the reachability matrix $R_r((A_k, B_k))$ at each $r \geq someinteger \mu$, and a nonsingular matrix T_r can be formed from those independent columns. Using the time-varying similarity in (4), a sequence $((A'_k, B'_k, C'_k, D'_k))$ can be eventually obtained. As mentioned in the previous section, however, the problem is how to find a 'canonical' algorithm that provides well defined invariants. Indeed, such canonical algorithm can be designed by considering the limit case of LPTV result as the period N goes to infinity. Note the following identity

$$[\lim_{N \to \infty} R(A_e, B_e)]T_c^\infty = diag\,(R_0((A_k, B_k)), R_1((A_k, B_k)), \ldots, R_k((A_k, B_k)), \ldots) \tag{27}$$

where T_c^∞ is an infinite dimensional column interchange (permutation) matrix. See the diagram in Fig. 3 associated with the reachability matrix $[\lim_{N\to\infty} R(A_e, B_e)]$. All r-numbered cells represent the reachability matrix $R_r((A_k, B_k))$ in (5). More specifically, a r-numbered cell $A_e^k[B_e]_{li}$ corresponds (is equal) to a column of $R_r((A_k, B_k))$ (with the infinite dimensional zero augmentation as in (27)) such as

$$A_e^k[B_e]_{li} \;\Leftrightarrow\; \overbrace{A_r A_{r-1} \cdots A_{r-k+1}}^{k}[B_l]_i \tag{28}$$

where the number r is determined by

$$r \;=\; k + l \tag{29}$$

Compare (28) and (29) with the LPTV case (23) and (24). The only difference is due to the modulo-N sense in (23) and (24). Now, a canonical form Γ^{LTV} : $((A_k, B_k, C_k, D_k)) \to ((A'_k, B'_k, C'_k, D'_k))$ is realized by the following algorithm:

Step (i): Let $r = 0$ and $\mu = 0$.
Step (ii): Search n independent columns in $R_r((A_k, B_k))$ with an order, see Fig. 3. If n independent columns are not found, repeat this step with the increment $r = r + 1$. Let $\mu = r$ only if n independent columns are found for the first time. Go to the next step.
Step (iii): Rearrange n independent columns with the fixed order, the scheme i, to form a nonsingular matrix T_r.

Step (iv): If $r = \mu$, then go to the step (ii) with $r = r + 1$. Otherwise, apply the time-varying similarity transformation in (4) to obtain the canonical qudruple (A'_r, B'_r, C'_r, D'_r). Go to the step (ii) with $r = r + 1$.

For the first example, suppose that with the search scheme *ii* as in Fig. 3 all the cells in the position $(1, 2, 1, 2, \ldots, 1, 2, \ldots)$ turn out to be dependent on the previous cells

$$\begin{aligned}
A_2[B_1]_1 &= a_{2,1}[B_2]_1 + a_{2,2}[B_2]_2 \\
A_2A_1[B_0]_2 &= a_{2,3}[B_2]_1 + a_{2,4}[B_2]_2 + a_{2,5}A_2[B_1]_2 \\
A_3[B_2]_1 &= a_{3,1}[B_3]_1 + a_{3,2}[B_3]_2 \\
A_3A_2[B_1]_2 &= a_{3,3}[B_3]_1 + a_{3,4}[B_3]_2 + a_{3,5}A_3[B_2]_2 \\
A_4[B_3]_1 &= a_{4,1}[B_4]_1 + a_{4,2}[B_4]_2 \\
A_4A_3[B_2]_2 &= a_{4,3}[B_4]_1 + a_{4,4}[B_4]_2 + a_{4,5}A_4[B_3]_2
\end{aligned}$$

The transformation matrices $T_1, T_2, \ldots$ are formed with the scheme *i* as in Fig. 3

$$\begin{aligned}
T_1 &= [[B_1]_1 \ [B_1]_2 \ A_1[B_0]_2] \\
T_2 &= [[B_2]_1 \ [B_2]_2 \ A_2[B_1]_2] \\
T_3 &= [[B_3]_1 \ [B_3]_2 \ A_3[B_2]_2]
\end{aligned}$$

Applying (4) to the above equations, the canonical sequence $((A'_k, B'_k))$ of pairs is for all $k \geq 1 + \mu$, $\mu = 1$

$$A'_k = \begin{bmatrix} a_{k,1} & 0 & a_{k,3} \\ a_{k,2} & 0 & a_{k,4} \\ 0 & 1 & a_{k,5} \end{bmatrix} \qquad B'_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{30}$$

For the second example, if with the scheme *ii* all the cells in the position $(1, 3, 2, 0, 0, 3$, $1, 3, 2, 0, 0, 3$, $\ldots, 1, 3, 2, 0, 0, 3, \ldots,)$ turn out to be dependent, then $\mu = 2$ and the canonical sequence $((A''_k, B''_k))$ is for all $k = 0, 1, \ldots$

$$\begin{aligned}
&A''_{3k+1+\mu} = \begin{bmatrix} a_{3k+3,1} & 0 & a_{3k+3,4} \\ a_{3k+3,2} & 1 & a_{3k+3,5} \\ a_{3k+3,3} & 0 & a_{3k+3,6} \end{bmatrix} \qquad A''_{3k+2+\mu} = \begin{bmatrix} a_{3k+4,1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&A''_{3k+3+\mu} = \begin{bmatrix} 1 & 0 & a_{3k+5,1} \\ 0 & 0 & a_{3k+5,2} \\ 0 & 1 & a_{3k+5,3} \end{bmatrix} \qquad B''_{3k+1+\mu} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \\
&B''_{3k+2+\mu} = \begin{bmatrix} 1 & a_{3k+4,2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad B''_{3k+3+\mu} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}
\end{aligned} \tag{31}$$

For the second example, if all the cells in $(1,2,2,2,3,0,1,2,0,5,2,2,0,0,2,2,\ldots)$ marked by ○'s in Fig. 3 are dependent, then $\mu = 1$ and the canonical sequence $((A_k''', B_k'''))$ is

$$A_{1+\mu}''' = \begin{bmatrix} 0 & 0 & a_{2,1} \\ 1 & 0 & a_{2,2} \\ 0 & 1 & a_{2,3} \end{bmatrix} \quad A_{2+\mu}''' = \begin{bmatrix} 0 & a_{3,1} & a_{3,4} \\ 1 & a_{3,2} & a_{3,5} \\ 0 & a_{3,3} & a_{3,6} \end{bmatrix} \quad A_{3+\mu}''' = \begin{bmatrix} 0 & 1 & 0 \\ a_{4,1} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A_{4+\mu}''' = \begin{bmatrix} a_{5,1} & 0 & a_{5,4} \\ a_{5,2} & 0 & a_{5,5} \\ a_{5,3} & 1 & a_{5,6} \end{bmatrix} \quad A_{5+\mu}''' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad A_{6+\mu}''' = \begin{bmatrix} a_{7,1} & a_{7,3} & 0 \\ a_{7,2} & a_{7,4} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

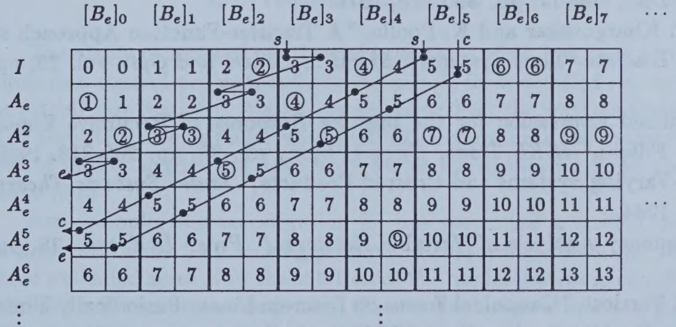
$$A_{7+\mu}''' = \cdots$$

$$B_{1+\mu}''' = \begin{bmatrix} 1 & a_{2,4} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad B_{2+\mu}''' = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \quad B_{3+\mu}''' = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$B_{4+\mu}''' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad B_{5+\mu}''' = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad B_{6+\mu}''' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$B_{7+\mu}''' = \cdots \tag{32}$$

The canonical pairs $(A_k', B_k'), (A_k'', B_k'')$ and (A_k''', B_k''') for $k \le \mu$ do not have specific structures.



	$[B_e]_0$		$[B_e]_1$		$[B_e]_2$		$[B_e]_3$		$[B_e]_4$		$[B_e]_5$		$[B_e]_6$		$[B_e]_7$	
I	0	0	1	1	2	②	3	3	④	4	5	5	⑥	⑥	7	7
A_e	①	1	2	2	3	3	④	4	5	5	6	6	7	7	8	8
A_e^2	2	②	③	③	4	4	5	⑤	6	6	⑦	⑦	8	8	⑨	⑨
A_e^3	3	3	4	4	⑤	5	6	6	7	7	8	8	9	9	10	10
A_e^4	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11
A_e^5	5	5	6	6	7	7	8	8	9	⑨	10	10	11	11	12	12
A_e^6	6	6	7	7	8	8	9	9	10	10	11	11	12	12	13	13

Fig. 3: the LTV Young diagram of the scheme *i* (as in 5- numbered cells) and the scheme *ii* (as in 3-numbered cells): $n = 3, m = 2$ and $N = \infty$ case

V. CONCULDING REMARKS : The LTV canonical form as in (30) of the structural invariant index $(1,2,1,2,\ldots\ ,\ 1,2\ ,\ \ldots)$, which is uniformly reachable, indeed generalizes respectively the LTI case (15) of the index $(1,2)$ and LPTV case (25) of $(1,2,1,2,1,2)$. In fact, the time-invariance in (12) and (13) and the periodicity in (16) and (17) allow one to obtain respectively the canonical quadruple in (15) and N-tuple in (25) using the LTV algorithm. Also, the LTV canonical form as in (31) of the structural index $(1,3,2,0,0,3\ ,\ 1,3,2,0,0,3,\ \ldots\ ,\ 1,3,2,0,0,3,\ldots)$ generalizes the

LPTV case as in (26) of the index $(1,3,2,0,0,3)$. A sequence as in (32) of the index $(1,2,2,2,3\ ,0,1,2,0,5,2,2\ ,0,0,2,2,\ldots)$ shows well a generally time-varying structure in LTV systems. The result [7] clarifies that the structural invariant index of LPTV system is constrained by the modulo property. For example, the so called modulo index $(1,0,0,1,2,2)$ of $(1,3,2,0,0,3)$ is certainly a permutation of $(0,0,1,1,2,2)$, see [7] for the details. It is an interesting open question what is simple expression of the constraint for the LTV case. A sequence $(1,2,2,3^*,3,0,1,2,0,5,2,2,0,0,2,2,\ldots)$, which is different from the above index on the digit with $*$, cannot be a structural invariant one of a LTV system $\sum_{n,m,p,\infty}^{cr}$.

Here, it would be emphasized that the significance of the canonical forms developed is in the theoretical understanding of LTV systems rather than in the numerical calculation. As mentioned in the introduction, the applications are expected in several areas of engineering and theory itself. In the future, it is to be investigated how the presented state- space description is related to the matrix transfer function description $H(z_1,z_2)$ as in [2][4] that is the usual one- sided z- transformation of the pulse response $h(k+r,r)$ of a LTV system over the two variables k and r. Because the two approaches are observed being reinforced by each other in the LTI case [9].

REFERENCES

[1] E.W. Kamen and K.M. Hafez, "Algebraic Theory of Linear Time-Varying Systems," *SIAM J. Contr. and Opt.*, vol. 17, pp. 500-510, 1979.

[2] E.W. Kamen, P.P. Khargonekar and K. Poolla, "A Transfer-Function Approach to Linear Time-Varying Discrete- Time Systems," *SIAM J. Contr. and Opt.*, vol. 23, pp. 550- 565, 1985.

[3] D. Li, "A Generalized Expression for the Impulse Response of Recursive Linear Time-Variant Digital Filters," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 205-218, 1989.

[4] J. Murray, "Time-Varying Systems and Crossed Products," *Math. Systems Theory*, vol. 17, pp. 217-241, 1984.

[5] L.A. Zadeh, "Frequency Analysis of Variable Networks," *Proc. IRE*, vol. 38, pp. 291-299, 1950.

[6] B. P. Park and E.I. Verriest, "Canonical Forms on Discrete Linear Periodically Time-Varying Systems and a Control Application," *IEEE Conf. Decision contr.*, Tempa, FL, 1989.

[7] B. P. Park, "Canonical Forms for Time-Varying Multivariable Linear Systems and Periodic Filtering and Control Applications," will be published, Ph. D. dissertation, School of Electrical Engineering, Georgia Inst. Tech., 1990.

[8] D. G. Luenberger, "Canonical Forms for Linear Multivariable Systems," *IEEE Trans. Automat. Contr.* AC-12, pp. 290-293, 1967.

[9] T. Kailath, *Linear System.* Englewood Cliffs, NJ: Prentice-Hall, 1980.

BLOCKING ZEROS AND DYNAMIC FEEDBACK*

MICHAEL K. SAIN and CHERYL B. SCHRADER
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

Blocking zeros represent one of the most intuitive ways of describing zero action in multivariable systems. Unfortunately, the discussion of such blocking activity can be complicated technically due to the system function failing to be one-to-one and onto. Moreover, additional subtleties occur when the system has decoupling zeros, either of input or output type. This paper presents a new theorem which permits the assessment of dynamical feedback upon blocking zeros, even when the system displays these complications, by means of defining open-loop equivalents of the key ideas involved. The interesting technical feature of the solution revolves around the relationship between the output decoupling zeros of the feedback system and those of the plant. For completeness, a similar analysis is presented for poles. The conversion of the feedback question to that of an equivalent open-loop question appears to be natural and promising for further investigation. Moreover, the solution possesses nuances which are not at all apparent from surface analyses.

I. INTRODUCTION AND PRELIMINARIES

The study of zeros of multivariable systems often has been at the center of investigations into control problem constraints; see [1] for a detailed survey of multivariable zeros and their applications. Among these researches, the module theoretic approach to pole and zero structure in such systems admits a coordinate free framework. Recently, a number of interesting results have been found which depend upon the concept of extended zeros [2-6].

For example, the investigation [6] has permitted a unified description of zero constraints in the model matching problem. It has been shown that there exists a module of fixed extended zeros Z in the solution of the model matching equation $T(z) = P(z)M(z)$. Moreover, if a solution exists, then there is a solution with extended zero module Z if and only if certain rank conditions on $M(z)$ and $T(z)$ apply. One of the central ideas in this investigation is the concept of Γ-zeros, extended zeros of Γ-type. With this generalized notion of zeros, Z is found to be an essential submodule of the extended, Γ-zeros of $P(z)$.

This idea of extended zero, together with an internal system view, is the context for the results reported herein. It is anticipated that generalized zero constraints will be found at the system level. A first step, then, is an investigation into the Γ-zeros in feedback systems. We utilize an approach dual to that of [4] in regard to extended zeros of Ω-type; and we examine for the first time in such a context the finite system poles.

We set the following conventions. Let k be an arbitrary field; then $k[z]$ is the ring of polynomials in z with coefficients in k, and $k(z)$ is its quotient field. Moreover, consider U, X, and Y, k-vector spaces of controls, states, and outputs having dimensions m, n, and p, respectively. Then we may employ the k-bilinear tensor product to form $k[z]$-modules

---

*This work was supported in part by the Frank M. Freimann Chair in Electrical Engineering at the University of Notre Dame, by the American Association of University Women Selected Professions Fellowship, and by the Zonta Amelia Earhart Fellowship.

$$\Omega U = k[z] \otimes_k U,\ \Omega X = k[z] \otimes_k X,\ \Omega Y = k[z] \otimes_k Y, \tag{1}$$

and $k(z)$-vector spaces

$$U(z) = k(z) \otimes_k U,\ X(z) = k(z) \otimes_k X,\ Y(z) = k(z) \otimes_k Y. \tag{2}$$

Because each of the modules in (1) is a $k[z]$-submodule of the corresponding space of rational vectors in (2), we denote the resulting factor modules by

$$\Gamma U = U(z)/\Omega U,\ \Gamma X = X(z)/\Omega X,\ \Gamma Y = Y(z)/\Omega Y. \tag{3}$$

Furthermore, let the state space maps $A : X \to X$, $B : U \to X$, $C : X \to Y$, and $D : U \to Y$ be k-linear; then $\Sigma(z) : X(z) \oplus U(z) \to X(z) \oplus Y(z)$ is a transfer function described by the Rosenbrock system matrix representation

$$\Sigma(z) = \begin{bmatrix} zI - A & B \\ -C & D \end{bmatrix}, \tag{4}$$

together with the associated system action $\Sigma(z)(x(z),\ -u(z)) = (0,\ -y(z))$. The system matrix (4) also defines two $k[z]$-module maps, $\Sigma_\Omega(z) : \Omega X \oplus \Omega U \to \Omega X \oplus \Omega Y$ and $\Sigma_\Gamma(z) : \Gamma X \oplus \Gamma U \to \Gamma X \oplus \Gamma Y$.

Corresponding to $\Sigma(z)$, there is a transfer function $G(z) = C(zI - A)^{-1}B + D$, a $k(z)$-linear map from $U(z)$ to $Y(z)$. Now the sequence

$$0 \to \Omega U \xrightarrow{i} U(z) \xrightarrow{G(z)} Y(z) \xrightarrow{p} \Gamma Y \to 0 \tag{5}$$

is exact at ΩU and ΓY, where again i and p are natural insertion and projection morphisms, respectively. A sequence is said to be exact at a module if the image of the incoming map equals the kernel of that outgoing. The Kalman input/output map associated with $G(z)$ is the $k[z]$-linear map $G^{\#} = p \circ G(z) \circ i$.

Beyond the usual transfer function, there exist other relationships between $\Sigma(z)$ and $G(z)$. Extend $B : U \to X$ to a $k[z]$-linear map $\widetilde{B} : \Omega U \to X$ by the action

$$\widetilde{B}\left(\sum_{j=0}^{\alpha} u_j z^j\right) = \sum_{j=0}^{\alpha} A^j B u_j. \tag{6}$$

In like manner, extend $C : X \to Y$ to a $k[z]$-linear map $\widetilde{C} : X \to \Gamma Y$ by the definition

$$\widetilde{C}x = \sum_{j=0}^{\infty} C A^j x z^{-j-1}\ mod\ \Omega Y. \tag{7}$$

Here, $im\ \widetilde{B}$ is the reachable subspace of X and $ker\ \widetilde{C}$ is the unobservable subspace of X. It then follows that $G^{\#} = \widetilde{C}(\Sigma)\widetilde{B}(\Sigma)$. If $\Sigma(z)$ is a minimal realization of $G(z)$, then $\widetilde{B}$ and $\widetilde{C}$ can be determined, up to an isomorphism, by $G(z)$, and we may write $\widetilde{B}(G)$ or $\widetilde{C}(G)$. For $x \in X$, zx is defined to be Ax.

In order to establish a third aspect of the transfer function, we choose also to discuss the exact $k[z]$-module sequence

$$0 \to \Omega U \to U(z) \xrightarrow{\pi_u} \Gamma U \to 0. \tag{8}$$

Let γ be an equivalence class in ΓU, and let u_γ be its strictly proper representative. Define the k-linear map $\sigma : \Gamma U \to U(z)$ by the action $\sigma(\gamma) = u_\gamma$; then $\pi_u \circ \sigma = id_{\Gamma U}$. Define also the k-linear map $\pi_y : Y(z) \to \Gamma Y$, the natural projection. With these definitions, we have the map $G_\# : \Gamma U \to \Gamma Y$ represented by $G_\# = \pi_y \circ G(z) \circ \sigma$.

Now consider the general feedback configuration of Figure 1, where $\Sigma(z)$ describes the plant at the system level and $\overline{\Sigma}(z)$ describes the compensator system. In a manner similar to (4), we can write

$$\overline{\Sigma}(z) = \begin{bmatrix} zI - \overline{A} & \overline{B}_r & \overline{B}_y \\ -\overline{C} & \overline{D}_r & \overline{D}_y \end{bmatrix} \tag{9}$$

with associated action $\overline{\Sigma}(z)(\overline{x}(z),\ -r(z),\ -y(z)) = (0,\ -u(z))$ for $\overline{\Sigma}(z) : \overline{X}(z) \oplus R(z) \oplus Y(z) \to \overline{X}(z) \oplus U(z)$. Likewise, the action equations for the feedback system of Figure 1 can be written in the form $\Sigma^{FB}(z)(\overline{x}(z),\ x(z),\ u(z),\ -r(z)) = (0,\ 0,\ 0,\ -y(z))$ for

$$\Sigma^{FB}(z) = \begin{bmatrix} zI - \overline{A} & -\overline{B}_y C & -\overline{B}_y D & \overline{B}_r \\ 0 & zI - A & -B & 0 \\ -\overline{C} & -\overline{D}_y C & -\overline{D}_y D + I & \overline{D}_r \\ 0 & -C & -D & 0 \end{bmatrix} \tag{10}$$

where $\Sigma^{FB}(z) : \overline{X}(z) \oplus X(z) \oplus U(z) \oplus R(z) \to \overline{X}(z) \oplus X(z) \oplus U(z) \oplus Y(z)$.

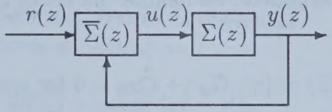


Figure 1

A compensator architecture which is more restrictive than that of Figure 1 is pictured in Figure 2, where

$$\overline{\Sigma}_H(z) = \begin{bmatrix} zI - \overline{A} & \overline{B}_y \\ -\overline{C} & \overline{D}_y \end{bmatrix} \tag{11}$$

with corresponding action $\overline{\Sigma}_H(z)(\overline{x}(z),\ -y(z)) = (0,\ -\overline{y}(z))$. Clearly, $\overline{\Sigma}_H(z) : \overline{X}(z) \oplus Y(z) \to \overline{X}(z) \oplus \overline{Y}(z)$. It is left to the reader to verify that the nomenclature of Figure 1 may reduce to that of Figure 2 for $\overline{B}_r = 0$.

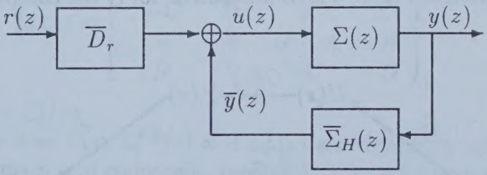


Figure 2

II. SYSTEMS AND MODULES

In this section we introduce the zero and pole modules of a system from an output point of view. Included in this discussion are the Γ-zero module of Σ, the zero module of blocked transmissions, the output decoupling zero module of Σ, and the system pole module.

We begin by examining the Γ-zero module of Σ, defined by

$$Z_\Gamma(\Sigma) = ker\ \Sigma_\Gamma(z). \tag{12}$$

The module of (12) is isomorphic to the direct sum of a finitely generated, torsion module of "lumped" zeros of the system and a torsion divisible module of zeros of the system. The former are of the same character as classical poles and the latter arise when the rank of $\Sigma(z)$ is not equal to the dimension of its domain. For example, this latter module vanishes when the system's corresponding transfer function is left invertible.

In order to discuss the concept of blocked transmissions, consider a strictly proper input $[u]$ representing an equivalence class in ΓU. What we are looking for are the strictly causal input sequences which can be blocked by an initial state. Now $[u]$ can be blocked by the output of a state $x_0 \epsilon X$ if $G_\#[u]+ \widetilde{C}x_0 = 0$, where we refer to map definitions of the preceding section. Moreover, we denote the set of all blockable signal equivalence classes by

$$Z_{signal}(\Sigma) = \{\gamma : G_\#\gamma + \widetilde{C}x_0 = 0 \text{ for some } x_0 \epsilon X\}. \tag{13}$$

The module of (13) is a $k[z]$-submodule of ΓU. The blocked transmission concept relayed herein is in the same spirit as the idea of blocked transmissions in Desoer and Schulman [7], but is at a system level.

A brief discussion of this transmission blocking philosophy is in order. Clarification is provided by means of an examination of previous results regarding minimal realizations [5]. Let

$$Z_\Gamma(G) = \frac{G^{-1}(\Omega Y) + \Omega U}{\Omega U} \tag{14}$$

be the Γ-zero module of $G(z)$. Define a $k[z]$-module map by $\widetilde{H} : Z_\Gamma(G) \to \Gamma U$, a natural inclusion. This map together with the module of (14) is called the "zero-signal generator" for $G(z)$. Furthermore, let $s : \Gamma U \to U(z)$ be a k-linear map which takes an equivalence class in ΓU to its unique strictly proper representative and let $\widetilde{H}_1 : Z_\Gamma(G) \to U(z)$ be $s \circ \widetilde{H}$. Then the diagram of Figure 3 is commutative, for $\widetilde{G}$ the composition $\pi \circ G(z) \circ \widetilde{H}_1$.

$$\begin{tikzcd}
& U(z) \arrow[r, "G(z)"] & Y(z) \arrow[dr, "\pi"] & \\
Z_\Gamma(G) \arrow[ur, "\widetilde{H}_1"] \arrow[rrr, "\widetilde{G}"] & & & \Gamma Y \\
& & X \arrow[ur, "\widetilde{C}"'] &
\end{tikzcd}$$

Figure 3

The above diagram admits the following interpretations. The image of $\widetilde{G}$ consists of outputs which correspond to the zero-signal generator. Now if $u(z) = \widetilde{H}_1(w)$ is a zero-signal, then $(\pi \circ G(z))u(z) = \widetilde{C}x$ for some x. Consequently, if the initial condition of $-x$ exists when the input $u(z)$ is applied, then the output is identically zero. Thus, this input $u(z)$ may be blocked by an initial state; and any strictly proper signal which can be so blocked is a zero-signal.

The output decoupling zero module,

$$Z_{od}(\Sigma) = ker \begin{bmatrix} zI - A \\ -C \end{bmatrix}_{\Gamma}, \tag{15}$$

is defined in a manner similar to Rosenbrock [8] as the set of unobservable states. It can be easily seen that the module of (15) is isomorphic to the $k[z]$-module $ker\ \widetilde{C}$.

The relationships between the three preceding system module definitions of (12), (13), and (15) can be summarized by the exact sequence [9]

$$0 \to Z_{od}(\Sigma) \to Z_{\Gamma}(\Sigma) \to Z_{signal}(\Sigma) \to 0. \tag{16}$$

It should be noted as a result of (16), that $Z_{\Gamma}(\Sigma)$ and $Z_{signal}(\Sigma)$ will be isomorphic as $k[z]$-modules if the system under consideration is observable.

Finally, the system pole module is defined as

$$P(\Sigma) = coker\ [zI - A]_{\Omega}. \tag{17}$$

The module described in (17) is both finitely generated and torsion.

The following section concentrates upon the Γ-zero module of a feedback system.

III. Γ-ZEROS IN FEEDBACK SYSTEMS

The section preceding has introduced modules in regard to single systems. The results of the present section establish a relationship between the Γ-zero module of a feedback system and its two main components: the output decoupling module of the plant system, and a specific module of transmission blocking zeros. We begin by examining the following lemmas.

Lemma A

Let $\Sigma_{\Gamma}^{1}(z) : \Gamma\overline{X} \oplus \Gamma R \oplus \Gamma X \oplus \Gamma U \to \Gamma\overline{X} \oplus \Gamma U \oplus \Gamma X \oplus \Gamma Y$ be given by

$$\Sigma_{\Gamma}^{1}(z) = \begin{bmatrix} zI - \overline{A} & \overline{B}_r & 0 & 0 \\ -\overline{C} & \overline{D}_r & 0 & I \\ 0 & 0 & zI - A & -B \\ 0 & 0 & -C & -D \end{bmatrix}_{\Gamma}; \tag{18}$$

then $ker\ \Sigma_{\Gamma}^{FB}(z) \cong ker\ \Sigma_{\Gamma}^{1}(z)$.

Proof: It can be easily seen that $\Sigma_{\Gamma}^{FB}(z)$ and $\Sigma_{\Gamma}^{1}(z)$ are similar by applying basis transformations to both domain and codomain, regarded as k-vector spaces.

We wish to relate modules on $\Sigma^{FB}(z)$ to those on $\Sigma(z)$ and $\overline{\Sigma}(z)$. To this end we use a block column partition. Specifically, we focus upon the mapping

$$\begin{bmatrix} 0 & 0 \\ 0 & I \\ zI - A & -B \\ -C & -D \end{bmatrix}_{\Gamma} : \Gamma X \oplus \Gamma U \to \Gamma\overline{X} \oplus \Gamma U \oplus \Gamma X \oplus \Gamma Y, \tag{19}$$

which we denote by $\Sigma_\Gamma^2(z)$.

Lemma B

Let $i_1 : \Gamma X \oplus \Gamma U \rightarrow \Gamma \overline{X} \oplus \Gamma R \oplus \Gamma X \oplus \Gamma U$ be the injection into the biproduct forming the domain for $\Sigma_\Gamma^1(z)$. Denote by I the identity homomorphism on the codomain of $\Sigma_\Gamma^1(z)$ to itself. Then the diagram of Figure 4 commutes.

$$\begin{tikzcd}
\Gamma X \oplus \Gamma U \arrow[r, "\Sigma_\Gamma^2(z)"] \arrow[d, "i_1"'] & \Gamma \overline{X} \oplus \Gamma U \oplus \Gamma X \oplus \Gamma Y \arrow[d, "I"] \\
\Gamma \overline{X} \oplus \Gamma R \oplus \Gamma X \oplus \Gamma U \arrow[r, "\Sigma_\Gamma^1(z)"] & \Gamma \overline{X} \oplus \Gamma U \oplus \Gamma X \oplus \Gamma Y
\end{tikzcd}$$

Figure 4

Proof: Choose a typical element $(x(z),\ u(z))$ in the domain of $\Sigma_\Gamma^2(z)$ and calculate its image under $I \circ \Sigma_\Gamma^2(z)$ and $\Sigma_\Gamma^1(z) \circ i_1$.

Lemma C

The sequence

$$0 \rightarrow ker\ \Sigma_\Gamma^2(z) \rightarrow ker\ \Sigma_\Gamma^1(z) \rightarrow \Gamma \overline{X} \oplus \Gamma R \overset{\Delta_\Gamma}{\rightarrow} coker\ \Sigma_\Gamma^2(z) \rightarrow coker\ \Sigma_\Gamma^1(z) \rightarrow 0 \qquad (20)$$

is exact, for an appropriately defined module map Δ_Γ.

Proof: See "snake lemma" [10]. Δ_Γ is called the "snake map".

We now look at the left portion of the sequence in Lemma C, which establishes that $ker\ \Sigma_\Gamma^2(z)$ is a submodule of $Z_\Gamma(\Sigma^{FB})$, by Lemma A. Then we proceed to place this Γ-zero module into an exact sequence.

Theorem 1

Let $\delta : ker\ [zI - \overline{A}\ \ \overline{B}_r]_\Gamma \rightarrow \Gamma U$ be the $k[z]$-linear map defined by the action

$$\delta(\overline{x}(z),\ r(z)) = [-\overline{C}\ \ \overline{D}_r]_\Gamma(\overline{x}(z),\ r(z)). \qquad (21)$$

Define the $k[z]$-module

$$Z_{signal}^r(\Sigma,\ \overline{\Sigma}) = \delta^{-1}\{im\ \delta \cap Z_{signal}(\Sigma)\}. \qquad (22)$$

Then the Γ-zero module $Z_\Gamma(\Sigma^{FB})$ may be placed in the exact sequence

$$0 \rightarrow Z_{od}(\Sigma) \rightarrow Z_\Gamma(\Sigma^{FB}) \rightarrow Z_{signal}^r(\Sigma,\ \overline{\Sigma}) \rightarrow 0 \qquad (23)$$

of $k[z]$-modules.

Proof: Omitted.

Clearly, the main components of $Z_\Gamma(\Sigma^{FB})$ are $Z_{od}(\Sigma)$ and $Z_{signal}^r(\Sigma,\ \overline{\Sigma})$. Further insight into the results of the above theorem can be achieved through an examination into the specialized nomenclature of Figure 2. Under these conditions, we have the resulting corollary.

Corollary I

If $\overline{B}_r = 0$ and $\overline{D}_r$ is a k-linear isomorphism, then the exact sequence of Theorem 1 becomes

$$0 \rightarrow Z_{od}(\Sigma) \rightarrow Z_\Gamma(\Sigma^{FB}) \rightarrow \delta^{-1}(Z_{signal}(\Sigma)) \rightarrow 0. \qquad (24)$$

Next, we examine some of the properties of $Z^r_{signal}(\Sigma,\ \overline{\Sigma})$. Special interest attaches to *ker* δ, which is just

$$Z^r_\Gamma(\overline{\Sigma}) = ker \begin{bmatrix} zI - \overline{A} & \overline{B}_r \\ -\overline{C} & \overline{D}_r \end{bmatrix}_\Gamma . \tag{25}$$

It is thus to be expected that $Z^r_\Gamma(\overline{\Sigma})$ is contained within $Z^r_{signal}(\Sigma,\ \overline{\Sigma})$. These thoughts can be presented in terms of an exact sequence also:

$$0 \to Z^r_\Gamma(\overline{\Sigma}) \to Z^r_{signal}(\Sigma,\ \overline{\Sigma}) \xrightarrow{\delta} Z^3 \to 0 \tag{26}$$

where we have denoted Z^3 by $im\ \delta \cap Z_{signal}(\Sigma)$.

IV. OUTPUT DECOUPLING ZEROS AND MODULES

In this section we propose to examine the output decoupling zero module of the entire feedback system and the manner in which it relates to the output decoupling zero module of the plant system. In general outline, this section is similar to the development up to and including Theorem 1 of the preceding section. Therefore, details are omitted in the interest of space.

The output decoupling zero module of the feedback system, $Z_{od}(\Sigma^{FB})$, can be studied from the kernel of

$$\begin{bmatrix} zI - \overline{A} & -\overline{B}_y C & -\overline{B}_y D \\ 0 & zI - A & -B \\ -\overline{C} & -\overline{D}_y C & -\overline{D}_y D + I \\ 0 & -C & -D \end{bmatrix}_\Gamma . \tag{27}$$

Up to k-linear isomorphism, we may as well examine

$$\begin{bmatrix} zI - \overline{A} & 0 & 0 \\ -\overline{C} & 0 & I \\ 0 & zI - A & -B \\ 0 & -C & -D \end{bmatrix}_\Gamma \tag{28}$$

which we denote by $\Sigma^3_\Gamma(z)$. Partitioning again by block column, we identify $\Sigma^2_\Gamma(z)$ once more, and obtain the commutative diagram of Figure 5.

$$\begin{tikzcd}
\Gamma X \oplus \Gamma U \arrow[r, "\Sigma^2_\Gamma(z)"] \arrow[d, "i_1"] & \Gamma\overline{X} \oplus \Gamma U \oplus \Gamma X \oplus \Gamma Y \arrow[d, "I"] \\
\Gamma\overline{X} \oplus \Gamma X \oplus \Gamma U \arrow[r, "\Sigma^3_\Gamma(z)"] & \Gamma\overline{X} \oplus \Gamma U \oplus \Gamma X \oplus \Gamma Y
\end{tikzcd}$$

Figure 5

The exact sequence

$$0 \to ker\ \Sigma^2_\Gamma(z) \to ker\ \Sigma^3_\Gamma(z) \to \Gamma\overline{X} \to coker\ \Sigma^2_\Gamma(z) \to coker\ \Sigma^3_\Gamma(z) \to 0 \tag{29}$$

follows again by the "snake lemma", where $ker\ \Sigma^2_\Gamma(z) \cong Z_{od}(\Sigma)$. The question remains as to how $ker\ \Delta_\Gamma$ has been changed by these steps.

Theorem 2

Define the $k[z]$-linear map $\overline{\delta} : ker\ [zI - \overline{A}]_\Gamma \rightarrow \Gamma U$ by

$$\overline{\delta}(\overline{x}(z)) = -\overline{C}\overline{x}(z); \tag{30}$$

define the $k[z]$-module

$$Z^{(r,y)}_{signal}(\Sigma,\ \overline{\Sigma}) = \overline{\delta}^{-1}\{im\ \overline{\delta} \cap Z_{signal}(\Sigma)\}. \tag{31}$$

Then the output decoupling zero module $Z_{od}(\Sigma^{FB})$ may be placed in the exact sequence

$$0 \rightarrow Z_{od}(\Sigma) \rightarrow Z_{od}(\Sigma^{FB}) \rightarrow Z^{(r,y)}_{signal}(\Sigma,\ \overline{\Sigma}) \rightarrow 0. \tag{32}$$

Proof: Omitted.

We turn now to a few remarks on the pole module.

V. FEEDBACK SYSTEM POLES

For study of $P(\Sigma^{FB})$, at the Γ-level, we consider the calculation of the kernel of

$$\Sigma^4_\Gamma(z) = \begin{bmatrix} zI - \overline{A} & -\overline{B}_yC & -\overline{B}_yD \\ -\overline{C} & -\overline{D}_yC & -\overline{D}_yD + I \\ 0 & zI - A & -B \end{bmatrix}_\Gamma . \tag{33}$$

Then we consider certain k-linear isomorphisms on the codomain of

$$\Sigma^5_\Gamma(z) = \begin{bmatrix} -\overline{B}_yC & -\overline{B}_yD \\ -\overline{D}_yC & -\overline{D}_yD + I \\ zI - A & -B \end{bmatrix}_\Gamma , \tag{34}$$

in particular an operation $(I - \overline{D}_yD)^{-1}$ on the second row, where we assume that $\Sigma^{FB}(z)$ is well posed. This yields

$$\begin{bmatrix} -\overline{B}_yC & -\overline{B}_yD \\ -(I - \overline{D}_yD)^{-1}\overline{D}_yC & I \\ zI - A & -B \end{bmatrix}_\Gamma . \tag{35}$$

By k-linear row and column operations, we convert this array to

$$\Sigma^6_\Gamma(z) = \begin{bmatrix} -\overline{B}_y(I - D\overline{D}_y)^{-1}C & 0 \\ 0 & I \\ zI - A - B(I - \overline{D}_yD)^{-1}\overline{D}_yC & 0 \end{bmatrix}_\Gamma . \tag{36}$$

Now, we can fit $\Sigma^4_\Gamma(z)$ and $\Sigma^5_\Gamma(z)$ into the diagram of Figure 6.

$$\begin{tikzcd}
0 \arrow[d] & & 0 \arrow[d] \\
\Gamma X \oplus \Gamma U \arrow[rr, "\Sigma_\Gamma^5(z)"] \arrow[d, "i_1"'] & & \Gamma \overline{X} \oplus \Gamma U \oplus \Gamma X \arrow[d, "I"] \\
\Gamma \overline{X} \oplus \Gamma X \oplus \Gamma U \arrow[rr, "\Sigma_\Gamma^4(z)"] \arrow[d, "p_1"'] & & \Gamma \overline{X} \oplus \Gamma U \oplus \Gamma X \arrow[d] \\
\Gamma \overline{X} \arrow[rr] \arrow[d] & & 0 \arrow[d] \\
0 & & 0
\end{tikzcd}$$

Figure 6

We can state the basic relationships which result as a theorem.

Theorem 3

Let $(I - \overline{D}_y D)$ be a k-linear automorphism, and let

$$\Sigma_\Gamma^7(z) = \begin{bmatrix} zI - A - B(I - \overline{D}_y D)^{-1}\overline{D}_y C \\ -\overline{B}_y(I - D\overline{D}_y)^{-1}C \end{bmatrix}_\Gamma ; \tag{37}$$

then the pole module $P(\Sigma^{FB})$, as expressed in $ker\ \Sigma_\Gamma^4(z)$, is described by the exact sequence

$$0 \to ker\ \Sigma_\Gamma^7(z) \to P(\Sigma^{FB}) \to ker\ \overline{\Delta}_\Gamma \to 0, \tag{38}$$

where $\overline{\Delta}_\Gamma$ is the $k[z]$-linear map in the exact sequence

$$0 \to ker\ \Sigma_\Gamma^5(z) \to ker\ \Sigma_\Gamma^4(z) \to \Gamma\overline{X} \overset{\overline{\Delta}_\Gamma}{\to} coker\ \Sigma_\Gamma^5(z) \to coker\ \Sigma_\Gamma^4(z) \to 0, \tag{39}$$

with kernel given by

$$ker\overline{\Delta}_\Gamma = \begin{bmatrix} zI - \overline{A} \\ -\overline{C} \end{bmatrix}_\Gamma^{-1} \left\{ \begin{bmatrix} zI - \overline{A} \\ -\overline{C} \end{bmatrix}_\Gamma \Gamma\overline{X} \cap \begin{bmatrix} -\overline{B}_y C & -\overline{B}_y D \\ -\overline{D}_y C & -\overline{D}_y D + I \end{bmatrix}_\Gamma ker[zI - A \quad -B]_\Gamma \right\}. \tag{40}$$

Proof: Omitted.

VI. CONCLUSIONS

The purpose of this paper is to investigate the pole and zero structure in feedback systems from an output point of view. In particular, the Γ-zero module of a feedback system is found to contain the output decoupling zero module of the plant system as a submodule and a certain "transmission blocking" module as a factor module. In like manner, the output decoupling zero module of the plant system is found to be a submodule of the output decoupling zero module of the entire feedback system. The corresponding factor module, then, is another "transmission blocking" module. Thus, it has been shown that the concepts of Γ-zeros, output decoupling zeros, and transmission signal blocking are intimately related in feedback system structures. Finally, an explicit description of the main components on the Γ-level of the feedback system pole module is provided by a short exact sequence of $k[z]$-modules. It should be noted that these results are achieved independently of any minimality or rank conditions.

Further research directions with regard to a Γ-level view of poles and zeros in feedback systems include areas of interest in which Γ-zeros are known to play a key role. In addition to the importance of Γ-zeros in the model matching problem, a relationship between module theoretic zero definitions and geometric control theory has been found for strictly proper systems using Γ-zeros [5]. In fact, a $k[z]$-module isomorphism between the transmission zero module and V^*/R^*, where V^* is the supremal $(A,\ B)$-invariant subspace contained in $ker\ C$ and R^* is the supremal $(A,\ B)$-controllable subspace contained in $ker\ C$, is induced from a projection of $Z_\Gamma(G)$ onto V^*. It is results such as these which have inspired the authors to analyze the Γ-zero structure of feedback systems.

REFERENCES

[1] C. B. Schrader and M. K. Sain, "Research on System Zeros: A Survey," *International Journal of Control*, Vol. 50, No. 4, pp. 1407-1433, October 1989.

[2] M. K. Sain and B. F. Wyman, "The Fixed Zero Constraint in Dynamical System Performance," in *Linear Circuits, Systems and Signal Processing: Theory and Application*, C. I. Byrnes, C. F. Martin, and R. Saeks, Editors. New York: North-Holland, 1988, pp. 87-92.

[3] M. K. Sain, B. F. Wyman, and J. L. Peczkowski, "Zeros in Plant Specification: Constraints and Solutions," *Proceedings 1988 American Control Conference*, Vol. 2, pp. 1243-1248, June 1988.

[4] M. K. Sain, C. B. Schrader, and B. F. Wyman, "A Theorem on the Effects of Compensator Poles on Feedback System Zeros," *Proceedings 1990 International Symposium on Circuits and Systems*, Vol. 2, pp. 1385-1388, May 1990.

[5] B. F. Wyman and M. K. Sain, "On the Zeros of a Minimal Realization," *Linear Algebra and Its Applications*, Vol. 50, pp. 621-637, 1983.

[6] M. K. Sain, B. F. Wyman, and J. L. Peczkowski, "Matching Zeros: A Fixed Constraint in Multivariable Synthesis," *Proceedings 27th Conference on Decision and Control*, pp. 2060-2065, December 1988.

[7] C. A. Desoer and J. D. Schulman, "Zeros and Poles of Matrix Transfer Functions and Their Dynamical Interpretation," *IEEE Transactions on Circuits and Systems*, CAS-21, pp. 3-8, 1974.

[8] H. H. Rosenbrock, *State-Space and Multivariable Theory.* New York: John Wiley & Sons, Inc., 1970.

[9] B. F. Wyman and M. K. Sain, "Module Theoretic Zero Structures for System Matrices," *SIAM Journal on Control and Optimization*, Vol. 25, No. 1, pp. 86-99, January 1987.

[10] M. F. Atiyah and I. G. MacDonald, *Introduction to Commutative Algebra.* Reading, MA: Addison-Wesley, 1969.

Computational Simplification in Controllability and Observability Tests

S. Bingulac
EE Department
Virginia Tech
Blacksburg, VA 24061

D. William Luse
The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102

Abstract: This paper considers the testing of controllability/observability of a linear time–invariant system by checking for fixed modes of a parameter–dependent matrix. It is shown that the order of the eigenvalue problems whose solution is required for this testing can be reduced by the number of linearly independent inputs/outputs of the system.

Introduction

It is well known that there are numerous procedures for checking the controllability and observability of state space representations of linear MIMO dynamic systems [e.g. 1,2,3]. Among the most popular are (i) calculation of ranks of controllability and observability matrices, (ii) similarity transformation into Jordan canonical form, (iii) Kalman canonical decomposition, (iv) transformation into Hessenberg form, (v) Popov–Belevitch–Hautus (PBH) tests, (vi) checking the existence of "fixed" closed–loop eigenvalues under feedback, etc. These procedures, being of different nature, each have unique properties, advantages, and disadvantages. Some are computationally ill–conditioned, and some require extensive computation. Others are not well suited to the case of multiple eigenvalues, since they require additional computation when multiple eigenvalues are present.

The purpose of this note is to give a simplified controllability/observability test. The test reduces to finding the eigenvalues of matrices whose order is less than that of the system by the number of non–redundant inputs/outputs. All of the steps are computationally straight–forward. In addition to information concerning controllability and observability, the test also gives information about the degree of controllability and observability.

Simplified Controllability/Observability Test

Consider an n–th order linear MIMO dynamic system with m inputs and p outputs represented by a state space representation $\{A,B,C\}$ that is not necessarily minimal. Without loss of generality, we assume that there are no redundant inputs and outputs, i.e., that: $\rho(B) = m$ and $\rho(C) = p$. The eigenvalues of A will be denoted by the set: $\lambda(A) = \Lambda = \{\lambda_i\}$, $i=1,...,n$.

Before stating the main result, we need the following definition.

Definition: Let $Q(X)$ be a k×k matrix–valued function of the r×s matrix X defined for almost every matrix X. A set of fixed eigenvalues q_j, $j \in \{1,...,k_1\}$ of $Q(X)$ is a set of eigenvalues of $Q(X)$ that is invariant with respect to arbitrary variations of the matrix X. That is: $q_j \in \lambda[Q(X)]$ for almost every X.

We can now prove the following result.

Theorem: Let the rows of N be a basis for the null space of B^T. Then the pair $\{A,B\}$ is controllable if an only if the following (k×k), k=n–m, matrix has no fixed eigenvalues with respect to arbitrary variations of the matrix X.

$$Q\,(X) = (N\,X)^{-1} N\,A\,X \quad \text{or} \quad Q\,(X) = N\,A\,X\,(N\,X)^{-1}$$

Proof: We may assume, without loss of generality, that the system is in Kalman controllable form because the effect on $Q\,(X)$ of a similarity transformation is:

$$Q\,(X) = N\,T^{-1}(TAT^{-1})\,X\,(N\,T^{-1}X)^{-1} = N\,A\,\tilde{X}\,(N\,\tilde{X})^{-1}, \text{ where } \tilde{X} = T^{-1}X.$$

The matrix $\tilde{X}$ is arbitrary because is is related to the arbitrary matrix X through an invertible transformation. Assuming now that the system is in Kalman form, we may write

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ 0 & 0 & A_{33} \end{bmatrix} \quad B = \begin{bmatrix} B_1 \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

where: A_{11} is m×m, A_{12} is $m\times(k-k_1)$, A_{13} is $m\times k_1$, A_{21} is $(k-k_1)\times m$, A_{22} is $(k-k_1)\times(k-k_1)$, A_{23} is $(k-k_1)\times k_1$, A_{33} is $k_1\times k_1$, and B_1 is m×m.

We assume that the size k_1 of the matrix A_{33} is as large as possible. Under this assumption, the pair $\{A_{22},A_{21}\}$ must be controllable, because the size of the matrix A_{33} could be increased through a similarity transformation that brings the pair $\{A_{22},A_{21}\}$ to Kalman form while not affecting any of the zero blocks in (1).

For a system in the form (1), the matrix N becomes:

$$N = \begin{bmatrix} 0 & I_{(k-k_1)\times(k-k_1)} & 0 \\ 0 & 0 & I_{k_1\times k_1} \end{bmatrix}$$

The matrix X is partitioned as: $X = |X_1^T|X_2^T|^T$, where X_1 is m×k, and X_2 is k×k.

We can now write: $$N\,A\,X\,(N\,X)^{-1} = \begin{bmatrix} A_{21} \\ 0 \end{bmatrix} X_1 X_2^{-1} + \begin{bmatrix} A_{22} & A_{23} \\ 0 & A_{33} \end{bmatrix} \tag{2}$$

The eigenvalues of (2) are clearly those of A_{33} together with those of a system $\{A_{22},A_{21}\}$ with an arbitrary feedback applied. As pointed out previously, this pair is completely controllable. Thus, the fixed eigenvalues of Q(X) are exactly the uncontrollable eigenvalues of the system {A,B}. This proves Theorem as well as the following Corollary.

Corollary: All fixed eigenvalues of the matrix Q(X) are equal to some eigenvalues of A, and they represent the uncontrollable eigenvalues of the pair {A,B}.

The proof of the observability of the pair {A,C} is dual of this proof.

Degree of Controllability

In the case that an eigenvalue of A is "almost" uncontrollable, it is natural to expect that some eigenvalue of Q(X) will be "almost" fixed, i.e., for various arbitrary matrices X, it will be located in the s–plane within a small circle about λ_i. Therefore, as the degree r_i of controllability of, one may define the radius of the smallest circle in the s–plane, centered at the eigenvalue λ_i, encompassing all locations where, for various arbitrary matrices X, the corresponding almost–fixed eigenvalue of Q(X) falls. Thus, the degree of controllability can be written as: $$r_i = \max_{X\in\mathbb{R}^{k\times n}} \{|\lambda_i - q_j(X)|\}$$

References

[1] T. Kailath, *Linear Systems*, Englewood Cliffs, NJ: Prentice–Hall, 1980.
[2] C. T. Chen, *Linear System Theory and Design*, New York: Holt, Rinehard and Winston, 1984.
[3] W. L. Brogan, *Modern Control Theory*, Englewood Cliffs, NJ: Prentice–Hall, 1982.

Computational Simplification in ERA (Eigensystem Realization Algorithm)

Bhaskar M. Gorti, S. Bingulac, H. F. VanLandingham
Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA , 24061

Abstract.

A computational simplification of the Ho-Kalman minimal realization procedure, recently referred to as the Eigensystem Realization Algorithm (ERA), is proposed. According to the simplified algorithm, instead of performing the singular value decomposition (SVD) of the Hankel matrix and multiplying with matrices containing left and right singular vectors, a minimal system state space representation may be obtained by simple selecting appropriate rows or columns from the Hankel matrix. The obtained state representation is in either Pseudo-Controllable or Pseudo-Observable form.

Introduction.

In their seminal paper [1], Ho and Kalman suggested a procedure for calculating a minimal state space representation $R=\left\{A,B,C\right\}$ of a MIMO system from given Markov Parameters Y_i , defined by $Y_i = CA^iB$; i = 0 , 1 , . . . , N. Since publication of that paper a number of papers and textbooks, [1]-[8], have referred to this minimal realization procedure and have suggested slightly modified versions. Relatively recently, this algorithm has been given considerable attention , particularly in the area of large flexible structures, where it has received a name ERA (Eigensystem Realization Algorithm), [9]-[12]. In all these papers different versions were suggested , but all of them basically follow the following procedure.

1: Define Markov Parameters , i.e. (p × m) matrices Y_i , of an n$^{\text{th}}$ order MIMO system with m inputs and p outputs.

2: Build the (rp × rm) Hankel matrix H_r given by, $H_r = \begin{bmatrix} Y_o .. Y_{r-1} \\ \vdots \quad \vdots \\ Y_{r-1} . Y_{2r-2} \end{bmatrix}$ for an arbitrary integer r.

3: Increase the r sequentially, until the rank of H_r does not increase, i.e when $\rho|H_r| = \rho|H_{r+1}|$, then n = $\rho\,|H_r|$ represents the order of a minimal state space representation.

4: Perform the SVD of H_r, i.e. calculate matrices U, Σ_n and V satisfying

$$H_r = U \begin{bmatrix} \Sigma_n \ldots O \\ \vdots \quad\; \vdots \\ O \ldots O \end{bmatrix} V^T \text{ where } \Sigma_n = \text{diag}\left\{\sigma_1, \ldots, \sigma_n\right\},\ \sigma_i > 0$$

5: The matrices A, B, C in a minimal realization are given by $A = (\Sigma_n)^{-1/2}\, U_1^T\, \widetilde{H}_r\, V_1\, (\Sigma_n)^{-1/2}$,

$B = (\Sigma_n)^{1/2}\, V_{11}$

where $\widetilde{H}_r = \begin{bmatrix} Y_1 \ldots Y_r \\ \vdots \quad \vdots \\ Y_r .. Y_{2r-1} \end{bmatrix}$ and U_1,V_1, U_{11} and V_{11} are (rp × n), (rm × n), (n × m) and (p × n) matrices respectively.

$C = U_{11}\, (\Sigma_n)^{1/2}$

U_1 contains the first n columns from U and V_1 contains the first n columns from V. Similarly,U_{11} contains the first p rows from U_1 and V_{11} contains the first m columns from $(V_1)^T$.

Simplified ERA.

The computational simplification in ERA suggested in this paper is based on the use of pseudo controllability/observability indices, introduced recently. Consequently, as explained in [13]-[16], there are more equivalent state space representations in pseudo controllable/observable forms corresponding to the given Markov parameters, Y_i. Thus, after calculating the rank of H_r,(steps 1,2,3 of the above algorithm), the only calculations that should be done, in the case of representing the system in a pseudo controllable form are : **(i.)** Select n appropriate columns from H_r with locations ν_1^c ,.., ν_n^c ; $\nu_i^c < \nu_{i+1}^c$, into the (rp × n) matrix H_1 and select n columns from H_r with locations m + ν_1^c ,.., m + ν_n^c , into the (rp × n) matrix H_2. **(ii.)** The system matrix A_c in a pseudo controllable form is given by the least square solution of

$$H_1 A_c = H_2 \quad\text{or}\quad A_c = (H_1^T\, H_1)^{-1} H_1^T\, H_2 \tag{1}$$

The input matrix B_c is fixed and is always given by $B_c = [\ I_m \vdots O\ ]^T$, while the output matrix C_c contains the first p rows from H_1. Similarly, in order to obtain a state space representation in a pseudoobservable form , the procedure is dual to the procedure for$\left\{A_c,B_c,C_c\right\}$. Obviously, the calculation required by (1), is much simpler than those required in the steps 4-5 of the original ERA algorithm. Particularly, when the system order is known or assumed calculation of SVD is not required. The n location numbers ν_1^c ,.., ν_n^c, defining the columns of H_r to be selected in matrices H_1 and H_2 are uniquely related to the assumed set of admissible pseudo controllability indices, respectively, [15].

This work is supported by the Office of Naval Research under Grant N00014-89-J-3123.

Example

Consider the example given in [17], pp 491-492:
Given Markov Parameters Y_i, i = 0, . . . , 5 : $\begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 7 \\ 1 & 2 \end{bmatrix}$

The order of the minimal realization and the number of inputs and outputs are n = 4, m = 2 and p = 2. The possible sets of pseudo controllable indices are {1,3},{2,2} and {3,1} . The Hankel matrix H and the locations $\{\nu_i^c\}$ corresponding to the sets of pseudo controllability indices are :

$$H = \begin{bmatrix} Y_0 \ldots Y_2 \\ \vdots \quad \vdots \\ Y_2 \ldots Y_5 \end{bmatrix} .$$

p.c.i :	{ 1, 3 }	{ 2, 2 }	{ 3, 1 }
locations :	{ 1, 2, 4, 6 }	{ 1, 2, 3, 4 }	{ 1, 2, 3, 5 }

The matrices H_1 and H_2 are formed using the above location indices, as defined in the earlier section. It is verified that the matrix H_1 in the case of p.c.i {2,2} is not of full rank, therefore, this set of p.c.i is not admissible . According to (1), the pseudo controllable forms $\{A_c, B_c, C_c\}$ corresponding to pseudo controllability indices {1,3} and {3,1} are :

p.c.i {1,3} : $$A_c = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 2 & 0 & 0 & 2 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad B_c = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad C_c = \begin{bmatrix} -1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

p.c.i {3,1} : $$A_c = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 2 & 0 & 2 \\ 1 & -1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad B_c = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad C_c = \begin{bmatrix} -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

References.

[1] Ho, B. L and Kalman, R. E., "Effective Construction of Linear State-Variable Models from Input/Output Data," *Proceedings of the 3rd Annual Allerton Conference on Circuit and System Theory,* 1965 pp. 449-459, also Regelungstechnik, vol. 14, 1966, pp. 545-548.

[2] Chen, Chi-Tsong, "*Linear System Theory and Design,*" 1984, *Holt, Rinehart and Winston, N.Y.*

[3]. Brogan, L . W., "*Modern Control Theory,*" 1974, *Quantum Publishers, Inc, N.Y.*

[4] Ackermann, J., "*Sampled - Data Control Systems*", Springler-Verlag, 1985.

[5] Sinha, N. K. and Kustza, B., "*Modeling and Identification of Dynamic Systems,*" 1984, Van Nostrand Rienhold.

[6] Gopinath, B., "On the Identification of Linear Time-Invariant Systems from Input/Output Data," *Bell System Tech J.,* vol. 48, 1969, pp. 1101 - 1113.

[7] Budin, M. A., " Minimal Realization of Discrete Linear Systems from Input/Output Observation" *IEEE Trans. .Automat. Contr.,* vol. AC-16, pp. 395-401, 1971.

[8] Silverman, L. M., " Realization of Linear Dynamical Systems ", *IEEE Trans. Automat. Contr.,* vol. ac-16, No. 6, 1971, pp. 554-567.

[9] Denman, E. et al., "Identification of Large Space Structures on Orbit", *Final Report,* AFRPL TR-86-054, Sept 1986.

[10] Juang, J. N. and Pappa, R. S., "An Eigensystem Realization Algorithm for Model Parameter Identification and Model Reduction," *Journal of Guidance, Control and Dynamics,* vol. 8, No. 5, Sept-Oct 1985, pp. 620-627.

[11] Pappa, R. S. and Juang, J. N., "Galileo Spacecraft Modal Identification Using an Eigensystem Realization Algorithm," *The Journal of the Astronautical Sciences,* vol. 33, no. 1, Jan-March 1985, pp. 15-33.

[12] *Proceedings of the NASA/JPL Workshop on Identification and Control of Flexible Space Structures,* San Diego, CA, June 4-6, 1984.

[13] Gevers, M. and Wertz, V., "Uniquely Identifiable State-space and ARMA parametrization for Multi-variable Linear Systems," *Automatica,* 20, 1982, pp. 333-347.

[14] Bingulac, S. and Krtolica, R. V., "An Algorithm for Simultaneous order and Parameter Identification in Multi-variable Systems," *IFAC -IFORS Symposium,* Beijing, PRC, pp.27-31, 1988.

[15] Bingulac, S. and Krtolica, R. V., "On Admissibility of Pseudoobservability and Pseudocontrollablity Indices," *IEEE Trans. Automat. Contr.,* 1987, vol. AC-32, pp. 920-922.

[16] Gorti, B. M., Bingulac, S. and VanLandingham, H. F., " Deterministic Identification of Linear Multi-variable Systems,". *Proceedings of the 22nd Southeastern Symposium on System Theory,* March 11-13, 1990, pp. 126-131 .

[17] Kailath, T., "*Linear Systems,*" 1980, Prentice - Hall, Inc.

A UNIFIED THEORY OF DISTRIBUTED COORDINATION

WITH COMMUNICATION UNCERTAINTY

Ray Strong Danny Dolev Flaviu Cristian

IBM Research
Almaden Research Center
650 Harry Road
San Jose, CA 95120

October 1, 1990

Abstract

A prominent piece of distributed systems folklore holds that safety and timeliness are incompatible properties for distributed coordination protocols. In this paper we provide a framework for understanding and comparing various lower bounds and impossibility results concerning the solution of coordination problems like atomic commit for distributed transactions and other consensus problems. Our treatment provides an abstract notion of information transfer that applies equally to systems with message based communication and systems with other information transfer mechanisms, such as shared memory. We provide a concise abstract definition of communication uncertainty that is independent of the particular information transfer mechanism.

1 Introduction

In this paper we focus on some of the key problems of fault tolerant distributed systems and what makes them hard to solve. We offer a unified theory that allows interpretation and comparison of many of the most important known limits on distributed systems. Among the demands placed on the designer of a distributed system, there is a competition between opposing demands for safety (consistency) and timeliness. We will argue that systems that require safety are generally subject to an isolation phenomenon that prevents timely coordination. We attempt to capture at a very high level of abstraction the possibility that faults or delays may force some processes in a distributed system to make decisions and take actions that are not entirely coordinated with those of other processes that remain functioning.

Our results are obtained in an abstract model of a distributed system that consists of a set of executions of the system together with an analysis of the system presented as a family of partial executions. Requirements for such an analysis are presented axiomatically. Any family that satisfies these requirements will serve as an analysis of the system; but some families provide a richer analysis than others.

Our results could be obtained using a knowledge oriented analysis based on possible executions in place of our families of partial executions (cf. [FHV, HM, HF, CM, PT]). However, the structures in such an analysis would be much more complex than those we introduce. The surprise is that we can obtain so much without resorting to knowledge based argument, relying only on a few simple axioms and definitions.

Our theory does not require a global concept of time, depending instead on a local ordering of events at each process in the distributed system. A novel contribution of our approach is the definition of "communication uncertainty" without the explicit mention of messages or other mechanisms for information transfer (cf. [L,DDS]).

The earliest discussed glimpse at this phenomenon of communication uncertainty, and one of the most intuitive, comes in the form of the Two Generals' Problem [G] (also known as the Chinese Generals' Problem). Two generals of allied armies are encamped on opposite hills above a valley occupied by the enemy. The generals' problem is to coordinate a time of attack. However, the only way they can communicate is by messenger through the valley, so there is uncertainty about whether any particular message will get through.

One of the most studied recent impossibility results in computer science is the result of [FLP] that it is impossible to reach consensus in an asynchronous system in the presence of at most one process crash failure, even when eventual communication is guaranteed. This result was generalized in [DDS] to apply to systems with various synchrony assumptions. The consensus problem is a coordination problem in the sense that all the processes that do not fail must agree on a common output. Multivalence is introduced because each process is given an input from a set of at least two values, and if all inputs agree, then all outputs must agree with the inputs. Note the similarity to the atomic commit problem (v. [BHG,G]) in which each process has some "input" that allows it to make a unilateral decision on whether to abort or allow commit; if all inputs allow commit and there are no failures, then the agreed on output must be commit. The essential distinction between the two problems is the constraint placed on processes recovering from failure in the atomic commit problem. The consensus problem is simpler because it puts no constraints on the actions taken by processes once they have failed. In both cases, communication is assumed to be uncertain. A fundamental characteristic of a communication uncertain system is that no information is conveyed to a process about the state of other processes by the number of actions the process takes. Using a new definition of communication uncertainty that attempts to capture this characteristic, we show that communication uncertain systems are vulnerable to isolation that can block the solution of any nontrivial coordination problem including consensus. Moreover, we show that, even in a model with guaranteed eventual communication and guaranteed eventual process repair, safety requires blocking. We extend this result to a generalization of the Fischer, Lynch, Paterson impossibility result. This generalization is expressed without reference to any specific mechanism for information transfer, such as messages, and allows immediate application to systems in which information is transferred by other mechanisms including shared memory

The reader may wish to compare our approach with alternate approaches (cf. [CM, PT, BMZ]).

The remainder of the paper is organized as follows: in Section 2 we provide our formal theory of distributed systems and give a general result relating safety and blocking for systems that solve nontrivial coordination problems; in Section 3 we provide a methodology for modelling distributed systems and prove the promised generalization of the [FLP] result.

2 Formal Model of Coordination.

We consider systems composed of at least two independently acting processes that can communicate with each other. The set of all possible process actions is denoted $\mathcal{A}$. During an execution of a system, each process has an infinite (ω ordered) sequence of events called the *local event sequence.* Each *event* e is composed of (1) an *id*, $id(e)$, consisting of the location at which the event takes place $@(e) \in \mathcal{P}$ (we identify a process with the location at which the events of its local event sequence take place), and a positive integer *sequence number* $s(e)$, (2) an *action*, $act(e) \in \mathcal{A}$, and (3) a *failure bit*, $f(e)$. Thus, in any execution, s is a function from events to positive integers, $@$ is a function from events to processes, and f is a function from events to $\{0,1\}$. An event e with $f(e) = 1$ is called a *failure event.* The function $@$ applied to any set of events will be the set of processes at which the events take place.

Formally, an *execution* is a set of events, such that projection on ids is a one-one mapping onto the Cartesian product of $\mathcal{P}$ (the set of processes) and the positive integers. We define a *distributed system* as a set of executions with at least two processes. We will use the term *system* as synonymous with the term *distributed system.*

A set of events is said to be *compatible* with respect to a system if it is contained in one of the executions of the system. When the particular system is clear from the context we will use the term "compatible" without mentioning the system. Two sets of events are said to be *compatible* if their union is compatible.

A *prefix* is a finite compatible set of events that is *closed under precedence* (if prefix x contains event e, then it contains all events at $@(e)$ with smaller sequence numbers than $s(e)$). A prefix may be thought of as a finite partial execution. We use incompatibilities between prefixes to capture the idea of information transfer within a distributed system.

A *coordination problem* for a distributed system consists of a subset $\mathcal{O}$ of $\mathcal{A}$ called *output actions,* a subset $\mathcal{R}$ of $\mathcal{P}$ called *participants* with cardinality at least two, and an equivalence relation $\equiv$ on $\mathcal{O}$ called *agreement.* A distributed system is said to *have a protocol* for a coordination problem $< \mathcal{O}, \mathcal{R}, \equiv >$ if, in every execution of the system, each process in $\mathcal{R}$ either fails or has at most one action from $\mathcal{O}$. (Formally, a *protocol* is a system together with a particular coordination problem for the system such that participants behave as described above in each execution.) A protocol is said to be *total* if, in each execution each process in $\mathcal{R}$ either fails or has exactly one action from $\mathcal{O}$. A protocol is said to be *safe* if, in each execution, all actions from $\mathcal{O}$ taken by processes in $\mathcal{R}$ that have not failed agree with respect to $\equiv$. When two output actions do not agree we will say they *disagree.* A coordination problem is said to be *nontrivial* for a distributed system if there are disagreeing output actions in the set of failure free executions of the system (the disagreeing outputs may be in different executions). Examples of nontrivial coordination problems include the atomic commit problem, the consensus problem, and the atomic broadcast problem. (For atomic broadcast, one participant is given an input value. If this participant does not fail, then all participants that do not fail must give the input value as output. In any case, all participants that do not fail must give the same value as output. Thus failure free executions of a protocol for the atomic broadcast problem must have disagreeing outputs if their inputs disagree.)

Our goal is to analyze our distributed system by choosing some family of prefixes to represent the "global states" or "cuts." The intuition leading to our axiomatization of such a family is contained in an example that assumes a Newtonian framework of real time. If each event in the system has associated with it a real time, and if events at one process can only "cause" or "influence" later events at other processes (no simultaneous action), then each prefix that corresponds to events that happen before some real time corresponds to a global state. We also add any subset of the events that happen at some particular real time to those that happen before. For each execution, the prefixes in the family are partially ordered by real time. This family of prefixes satisfies the first three axioms discussed below: Closure, Compatibility, and Separability. It turns out that for many purposes, we can drop the assumption of real time and simultaneity and obtain the same results by constructing another family of prefixes that satisfies the first two of these axioms. Moreover, we can characterize the intutive concept of "communication uncertainty" with the second pair of axioms below: Separability and Autonomy.

If x and y are sets of events, we denote the union of x and y by $x + y$; and, the set of events in x but not in y by $x - y$. Thus $@(y - x)$ is the set of locations of events in y but not in x, which may not be the same as $@(y) - @(x)$. A family of sets of events is said to be *closed under compatible union and intersection* if, whenever x and y are in the family and $x + y$ is compatible, then $x + y$ and $x \cap y$ are in the family. A family of sets of events is said to *cover* the set of prefixes of a distributed system if, for each prefix x and execution z, if x is contained in z, then there is a member y of the family that contains x and is contained in z. Prefix y

is said to be a *failure conserving* extension of prefix x if the only failure events in $y-x$ occur at processes that failed in x. A failure conserving extension y of x is a *strict* extension if the failure bit of any event in $y-x$ is the same as the failure bit of the last event in x at the same process.

An *analysis* of a distributed system is a family of prefixes (called *cuts*) that satisfies the following two axioms:

Closure Axiom: The family contains the empty prefix, covers the set of prefixes of the system, and is closed under compatible union and compatible intersection.

Compatibility Axiom: If x, y, and z are cuts with $x \subset y$ and $x \subset z$, and $@(y-x)$ and $@(z-x)$ are disjoint, then y and z are compatible.

The cuts are to be viewed as global states of our system; however, in general in our model, there is no notion of time or simultaneity, so cuts should not be considered to represent any notion of concurrency. Every system possesses at least one analysis. If a system possesses an analysis satisfying the following two additional axioms then we say the system has *communication uncertainty:*

Separability Axiom: For any pair of events at different processes that are contained in a cut, there is a compatible cut that contains one event but not the other.

Autonomy Axiom: For each cut x and process p, there is a cut y containing x such that $@(y-x)$ contains only p and y is a failure conserving extension of x.

It is communication uncertainty rather than any particular form of asynchrony that leads to impossibility results like those of [FLP]. Let the asynchronous message based systems of [FLP] (with no restriction on failure types or numbers) with analysis consisting of the family of prefixes that contain the corresponding message send event for every contained message receive event be called *standard message based systems.* It is easy to check that this analysis satisfies all four axioms above. (cf. [L]).

The results in the remainder of this section assume communication uncertainty. In the next section, an analysis that satisfies the separability axiom will be called *separable* and an analysis that satisfies the autonomy axiom will be called *autonomous.*

Cut y is said to be a *successor* of set of events x when $x \subset y$ and there is no cut z such that $x \subset z \subset y$. The transitive closure of the successor relation is denoted *leads to.* When x leads to y, y is said to *extend* x. When x leads to y and $|@(y-x)| = 1$, we say x leads to y *locally* and y is a *local extension* of x. Note that a successor to a set of events must be a cut and therefore a prefix.

If cut x has incompatible successor cuts y and z, then the triple $< x, y, z >$ is said to be a *fork.* The fork $< x, y, z >$ is said to be *local* if $@(y-x) \cup @(z-x)$ consists of one process. Fork $< x, y, z >$ is said to be *failure free* if y and z are failure free.

Lemma 2.1.: If cut y is a successor of cut x, then there is exactly one process in $@(y-x)$.

Proof: Suppose cut y were a successor of cut x and $@(y-x)$ contained more than one process. By the separability axiom, since $y-x$ contains events at different processes, there is a cut z compatible with y that contains some event of $y-x$ but does not contain some other event of $y-x$. By the closure axiom, the intersection of y and z is a cut properly contained in y but containing some event of $y-x$. Now x is also a cut properly contained in y, so the union w of x with the intersection of y and z is a compatible set of events and thus a cut. But w properly contains x and is properly contained in y; so y could not be a successor of x.
■

Lemma 2.2.: Every fork is local.

The proof is omitted.

A sequence $\{x(i)\}$ of prefixes is called an *ascending chain* if, for each i, $x(i) \subset x(i+1)$. A protocol is called *blocking* if there is an infinite ascending chain $\{x(i)\}$ such that some participant takes infinitely many actions in $\cup\{x(i)\}$ without failing or giving output. We say a protocol is *timely* if there exists an integer k such that, in every execution, every participant either fails or gives an output before it has taken k actions. Thus if a protocol is blocking, then it is not timely (cf. "wait-free" in [H]). Moreover, if a protocol is timely, then it is total.

Consider some fixed protocol. A cut is said to be *multivalent* if it is contained in failure free executions with disagreeing outputs. If a cut is contained in some failure free execution and if all outputs in the failure free executions that contain it agree, then the cut is said to be *univalent*. Note that if a system has a nontrivial coordination problem, then the empty cut is multivalent.

Theorem 2.3. : Any safe protocol for a nontrivial coordination problem in a communication uncertain system must be blocking.

Proof Sketch: The empty cut is multivalent. Either there is an infinite ascending chain of multivalent cuts or there is a multivalent cut with no multivalent successors. If there is an infinite ascending chain of multivalent cuts, then the protocol is blocking. If there is a multivalent cut x with no multivalent successors, then there must be a failure free fork $< x, y, z >$. By Lemma 2.2, there is a process p not contained in $@(y-x) \cup @(z-x)$. Now repeated application of the autonomy axiom yields an infinite ascending chain of failure free cuts, beginning with x, in which only p takes actions after x. By the compatibility axiom, each of these cuts must be compatible with both y and z, so p can neither fail nor give output in any of them. Thus, in any case, the protocol is blocking.
■

Corollary 2.4. : There is no safe timely protocol for a nontrivial coordination problem in a communication uncertain system.

3 Production of Models of Distributed Systems by Restriction

3.1 Syntax and Environment

Our method for constructing models of systems starts with a syntactically defined free algebra of executions. By restriction we introduce what we call an environment. This is a set of executions that contains in some sense executions for all possible protocols that could be written for the system. Then we further restrict the set of executions to those that correspond to the actions (state transitions) of a fixed set of automata. All our restrictions are specified as properties of sets of events, independent of any consideration about what system is being restricted. Thus an execution either satisfies or does not satisfy a restriction, independent of the system from which it is taken. Formally, a *restriction* is a property of sets of events. Applying a restriction ρ to a system S produces the set of executions ρS that satisfy the property ρ.

To model each example, we start with what we call a *syntactic* system consisting of all executions that can be formed from given sets of processes and actions. we apply restrictions to the syntactic system to obtain an environment, where an *environment* is a system E such that, for any execution of E, changing any failure bit produces another execution of E. Note that under this definition, a syntactic system is an environment. A restriction is said to be *environmental* in system S if the restriction is independent of which events are failure events for executions in S. For environments, the failure bits of events are irrelevant. We use the failure bits of events in a system to indicate events that are not according to the specifications of the protocols of the system but are still possible within an environment that contains the system.

Note that if E is an environment and ρ is an environmental restriction in E, then ρE is also an environment.

We can think of an environment as containing many potential protocols. To specify a single protocol, we restrict the environment to a system in which the actions correspond to state transitions of a given set of automata and the events with actions not specified by the automata are the failure events (cf. IO automata [LM,LT]).

3.2 Completeness and Continuity

A system is said to be *complete* if the union of the events of any ascending chain of prefixes is contained in an execution. Note that a complete system cannot have a total blocking protocol for a coordination problem.

An example of a restriction that does not preserve completeness is *eventual message delivery*. This restriction holds of a set of events if for each message sent the set contains a corresponding receive event or a failure event at the target process. Thus, if a system has eventual message delivery, then an infinite ascending chain of cuts in which some message is never delivered while its target takes infinitely many actions without failing is not contained in any execution of the system.

The following result follows immediately from Theorem 2.3.

Corollary 3.1.: (Generalization of Two Generals Problem) There is no safe total protocol for a nontrivial coordination problem in a complete communication uncertain system.

For each system S and restriction ρ, we define the restriction $ext_S(\rho)$ to hold of a set of events if that set can be extended to an execution of S that satisfies ρ. In other words, $ext_S(\rho)(x)$ if, and only if, x is a compatible set of events in ρS. A restriction ρ is said to be *continuous* on system S if the restriction $ext_S(\rho)$ is inherited by the union of any ascending chain of prefixes for which it holds. The following lemmas are immediate from the definitions.

Lemma 3.2.: System ρS is complete if, and only if, ρ is continuous on S.

Lemma 3.3.: If restrictions ρ and σ are continuous on system S then $(\rho|\sigma)$ is continuous on S. If, in addition, $ext_S(\rho) \wedge ext_S(\sigma)$ implies $ext_S(\rho \wedge \sigma)$, then $(\rho \wedge \sigma)$ is also continuous on S.

If prefix z can be obtained from prefix y by changing only failure bits, then z and y are said to be *similar*. A restriction ρ is said to *impose a protocol* on system S if for every prefix or execution y of S, there is exactly one prefix or execution of ρS that is similar to y.

Lemma 3.4.: If restriction ρ imposes a protocol on complete system S, then ρ is continuous on S.

The proof is omitted.

When we restrict an environment to executions with failure bits determined by a given set of automata (with state transitions corresponding to the actions of the environment), the restriction imposes a protocol on the environment. Thus such a restriction preserves completeness.

3.3 Special Restriction Types

We focus on two types of restrictions and the system properties that they preserve. The types are called "pattern enforcing" and "coherent." Each type is defined in this section.

If ρ is a restriction and R is a set of processes, then ρ is said to have *base* R if ρ is determined by the projection of its argument on the set of events that take place at processes in R. Here we define a specific type of restriction that can be used to state a powerful result for uncertain systems. Restriction ρ is said to be *basic* if

(1) it is inherited by supersets,
(2) it is inherited by some finite subset, and
(3) it has a base other than the set of all processes.

Messages provide a special case of what we call a basic pattern. The pair $< \alpha, \beta >$ of basic restrictions is a *basic pattern* for an analysis A if

(1) for each $x \in A$, $\beta(x)$ implies $\alpha(x)$;
(2) for each $x \in A$, $\alpha(x)$ implies x has a failure conserving local extension y in A such that $\beta(y)$; and
(3) there is a base for β consisting of a single process.

When a restriction can be expressed as the countable conjunction of implications of the form $(\alpha \Rightarrow \beta)$, where $< \alpha, \beta >$ forms a basic pattern for analysis A, we call the restriction *pattern enforcing* with respect to A. If "failure conserving" is replaced by "strict" in (2) above, then we call the restriction *strict pattern enforcing* with respect to A.

The restriction to "eventual message delivery" is a strict pattern enforcing restriction for standard message based systems. Each basic pattern $< \alpha, \beta >$ corresponds to the send and receive of a particular message. In general restrictions that specify eventual communication are usually pattern enforcing. However, restrictions that specify communication within a bounded number of actions (e.g. if the communication is initiated before action k of the sender, then receipt must take place before action $2k$ of the receiver) are usually not pattern enforcing because they cannot be expressed in terms of patterns that satisfy (2) above.

Lemma 3.5.: If S is complete and ρ is pattern enforcing with respect to some analysis of S, then any analysis for S forms an analysis for ρS. Moreover, any separable or autonomous analysis for S forms a respectively separable or autonomous analysis for ρS.

The proof is omitted.

A property ρ is said to be *pattern enforcing* for a system if it is pattern enforcing for some analysis of the system.

Corollary 3.6. : If restriction ρ is pattern enforcing for a complete communication uncertain system S, then ρS is communication uncertain.

One very strong restriction that is sometimes assumed in modelling distributed systems is the property of *eventual repair with immunity to failure* (after some point each process that has failed is repaired and has no subsequent failure). A weaker alternative that is somewhat more realistic is what we call *eventual repair* (for any integer k and any process p, there is some point after which p takes k actions without failing). The restriction to "eventual repair" is a pattern enforcing restriction for standard message based systems. For each basic pattern $< \alpha, \beta >$, we take $\alpha \equiv \mathbf{true}$. For each process p and integer k there is a corresponding β that requires p to have at least k contiguous events that are not failures in its local event sequence.

A restriction ρ is said to be *coherent* for analysis A of system S if it satisfies the following conditions:

(1) the empty prefix satisfies $ext_S(\rho)$,
(2) $ext_S(\rho)$ is inherited by compatible unions of members of A, and
(3) $ext_S(\rho)$ is inherited by strict extensions.

We say a restriction is *coherent* for a system if it is coherent for some analysis for the system.

If A is an analysis for system S and ρ is a restriction, then we define ρA to be the family of prefixes of ρS that are members of A.

Lemma 3.7.: If restriction ρ is coherent for analysis A of system S then ρA forms an analysis for ρS. Moreover, ρ preserves separability and autonomy.

The proof is omitted.

Let $\mathcal{R}$ be any subset of the set of processes. Let ρ be the restriction that is satisfied when failure events occur only at $\mathcal{R}$. Let S be any system that contains an execution satisfying ρ. Then ρ is coherent for S.

Theorem 3.8.: Suppose A is a separable autonomous analysis for a complete system S. Let ρ be any pattern enforcing restriction for A in S. Let τ be the restriction of having no process failures. Then $(\rho \wedge \tau)S$ has communication uncertainty.

Proof Sketch: Since τ is coherent, τA forms a separable autonomous analysis for τS by Lemma 3.7. Note that τ is continuous on S so that τS complete. Thus by Lemma 3.5 it suffices to show that ρ is pattern enforcing for τA in τS. But this is straightforward since a failure conserving extension of a failure free cut must also be failure free.

■

Corollary 3.9.: Let S be the restriction of a standard message based system to executions with any combination of eventual message delivery, eventual repair, and no process failures. Then S has no safe timely protocol for a nontrivial coordination problem.

Proof Sketch: The Corollary follows from Corollary 2.4 and Theorem 3.8 for the conjunction of all three restrictions.

■

Lemma 3.10.: If ρ is a strict pattern enforcing restriction for analysis A of system S and σ is a coherent restriction for A, then ρ is a strict pattern enforcing restriction for analysis σA of system σS.

The proof is omitted.

3.4 Crash Failures

A process is said to suffer a *crash failure* with respect to analysis A if its local sequence of events consists of failure free events followed by an infinite number of special events called *stopped* events such that if a is some last stopped event in a member x of A, then either the removal of a or the addition of a next stopped event provides another member of A. A system is said to be *subject to crash failures* with respect to analysis A if, for each process p, any prefix x is contained in a prefix y such that $y - x$ is a stopped event at p with respect to A. Note that the restriction of having only crash failures is continuous on any complete system that is subject to crash failures. Also, having only crash failures is inherited by failure conserving extensions in any system that is subject to crash failures and has only stopped events as failure events.

Theorem 3.11. : Suppose A is a separable autonomous analysis for a complete system S such that S is subject to crash failures and has only stopped events as failure events with respect to A. Let ρ be any pattern enforcing restriction for A in S; let σ be the restriction of having only crash failures; let τ be the restriction of having at most one failure; and let π be the conjuction $\rho \wedge \sigma \wedge \tau$. Then πS is a system that contains no safe total protocol for a nontrivial coordination problem.

Proof Sketch: (This proof uses techniques adapted from [FLP].) Assume the hypotheses of the theorem and assume that the conclusion is not satisfied because πS has a safe total protocol for a nontrivial coordination problem. We will write A' for $(\rho \wedge \sigma)A$ and S' for $(\rho \wedge \sigma)S$. By Lemmas 3.5, 3.7, and 3.10, A' is a separable autonomous analysis of S'.

Let $\mu(p)$ be the restriction of having failure events only at process p. Let ν be the restriction of having no failure events. Then $\mu(p)$ and ν are both coherent restrictions for A' in S'. Thus by Lemma 3.7, $\mu(p)A'$ and $\nu A'$ are separable autonomous analyses of their respective systems.

The triple $< x, y, z >$ of members of A' will be said to form a *local disagreement* if x is multivalent and leads to disagreeing univalent y and z with only one process in $@(y - x) \cup @(z - x)$. We will show that S' can have no local disagreements. Suppose that $< x, y, z >$ were a local disagreement at process p. Since they are all failure free, x, y, and z are all prefixes of $\mu(p)S'$. Since $\mu(p)S'$ is a subset of the executions of πS, each execution of $\mu(p)S'$ must contain output for all processes except possibly p. There must be some execution of $\mu(p)S'$ that contains x and in which p crashes immediately after its last event in x. For this execution, some process must give an output; so there must be some member of $\mu(p)A'$ that is univalent and has no event at p except for those of x. Since $\mu(p)A'$ forms an analysis for $\mu(p)S'$, this univalent cut must be compatible with both y and z, contradicting the safety of the hypothesized protocol. Thus S' can have no local disagreements.

Since S has a total protocol for a nontrivial coordination problem, the empty cut of $\nu A'$ is multivalent in $\nu S'$. We can now construct an infinite ascending chain of multivalent cuts in which each process takes infinitely many actions without failing. The construction iterates $n+1$ stages where n is the number of processes. In stages that are $i \bmod (n+1)$ with $0 < i < n+1$, we construct a cut containing the cut we have together with some new action of process i. In stages that are $0 \bmod (n+1)$, if ρ holds of the cut we have, then we do nothing; otherwise, we construct a failure free containing cut that satisfies the first implication $\alpha_i \Rightarrow \beta_i$ not satisfied by the original cut.

Using the key technique of [FLP], it is straightforward to show that if, at some stage, we cannot produce a multivalent cut with the required property, then there must be a local disagreement. Thus we can construct the desired infinite ascending chain. By construction the chain must satisfy property ρ. Since the chain has infinitely many actions at each process, its union is a failure free execution in πS. But this contradicts the assumption that the protocol is total in πS.

■

Corollary 3.12.: (FLP) Let S be the restriction of a standard message based system to executions with eventual message delivery, no failures except crash failures, and at most one process failure. Then S has no safe total protocol for a nontrivial coordination problem.

4 Conclusion.

When communication is uncertain, isolation is possible and there is no way to guarantee safety and timely (nontrivial) coordination among distributed processes. Usually the probability of isolation is small even though some components may be inoperative for long periods of time. Blocking prevents transient failures from causing a loss of consistency during isolation. Much as we would like a truly nonblocking total coordination protocol, Corollaries 3.1 and 3.6 imply that neither eventual message delivery nor eventual repair is enough to guarantee safety without blocking in an uncertain environment. We suggest that rather than search for real systems without the problems of communication uncertainty, we should search for systems in which isolation has low probability and choose either consistency or timeliness to guarantee, relegating the other goal to highly probable but not certain.

5 Acknowledgements

The authors would like to thank Joe Halpern for helpful comments on an earlier version of this work.

References

[BHG] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrent Control and Recovery in Database Systems,* Addison-Weseley, 1987.

[BMZ] O. Biran, S. Moran, and S. Zaks, "A Combinatorial Characterization of the Distributed Tasks Which are Solvable in the Presence of one Faulty Processor." *Proc. 7nd ACM Symp. on PODC,* (1988) 263-273.

[CM] K. M. Chandy, and J. Misra, "How Processes Learn." *Distributed Computing,* (1986) 1:40-52.

[DDS] D. Dolev, C. Dwork, and L. Stockmeyer, "On the Minimal Synchronism Needed for Distributed Consensus," *Journal of the ACM,* (1987) 34:77-97.

[FHV] R. Fagin, J. Y. Halpern, M. Y. Vardi, "A Model-Theoretic Analysis of Knowledge," *Proc. 25th IEEE Symp. on Foundation of Computer Science,* (1984) 268-278.

[FLP] M. J. Fischer, N. A. Lynch and M. S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *JACM 32* (1985) 373-382.

[G] J. N. Gray, "Notes on Database Operating Systems," *Operating Systems: an advanced course, Lecture Notes in Computer Science 60,* Springer Verlag (1978) 393-481.

[HF] J. Halpern and R. Fagin, "A Formal Model of Knowledge, Action, and Communication in Distributed Systems," *Proc. 4th ACM Symp. on PODC,* (1985) 224-236.

[HM] J. Y. Halpern, and Y. Moses, "Knowledge and Common Knowledge in a Distributed Environment," *Proc. 3rd ACM Symp. on PODC,* (1984).

[H] Maurice P. Herlihy, "Impossibility and Universality Results of Wait-Free Synchronization," *Proc. 7nd ACM Symp. on PODC,* (1988) 276-290.

[L] L. Lamport, "Time Clocks and the Ordering of Events in a Distributed System," *CACM 21,*7 (1978) 558-565.

[LM] N. A. Lynch, and M. Merritt, "Introduction to the Theory of Nested Transactions," *International Conf. on Database Theory,* Sep. 1986, Rome, Italy, 278-305. Technical Report MIT/LCS/TR-367, MIT, Lab. for Computer Science, April 1986.

[LT] N. A. Lynch, and M. R. Tuttle, *Hierarchical Correctness Proofs for Distributed Algorithms,* Technical Report MIT/LCS/TR-387, MIT, Lab. for Computer Science, April 1987.

[PT] P. Panangaden and K. Taylor, "Concurrent Common Knowledge: A New Definition of Agreement for Asynchronous Systems," *Proc. 7nd ACM Symp. on PODC,* (1988) 197-210.

Loosely Coupled Processes

Jayadev Misra*
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712
(512) 471-9547
misra@cs.utexas.edu

October 4, 1990

1 Introduction

Processes that communicate through shared variables are often called "tightly-coupled" whereas message communicating processes are usually termed "loosely-coupled." From a programming viewpoint, message communicating processes have a number of attractive features: They are easier to design (compared to tightly-coupled systems), by using program decomposition around the component processs, and they are easier to understand, by asserting facts about the transmitted message sequences. One of the key attributes of a message communicating system is that it obeys certain monotonicity conditions: A message that has been sent cannot be unsent nor can a received message be unreceived. Therefore, any assertion about a sequence that holds initially and that cannot be falsified by extending the sequence, holds for all message sequences sent and received.

Tightly-coupled systems, by contrast, require a deeper analysis for their understanding. In many cases, explicit protocols are required to guarantee that "race conditions" are avoided, i.e., independent of the speeds of component processes, shared variable values are appropriately written and read. Several programming constructs—semaphore, critical region, monitors—have been proposed to simplify programming with shared variables.

We show in this paper that many of the advantages of message communication can be realized in a tightly-coupled system if the processes access shared variables in a disciplined manner. These access conditions are generalizations of the monotonicity conditions for loosely-coupled systems alluded to previously. If these access conditions are met then each shared variable can be implemented as follows: Each process has a local copy of each shared variable it accesses; reads/writes are performed on the local copies instead of on the corresponding shared variables; whenever a change is made on a local copy, the change is communicated to each process that has a copy of this shared variable; each process updates its local copy appropriately upon receiving a notification of a change. At any time the different local copies of a shared variable may have different values, but we

---

*This work was partially supported by ONR Contract N00014-90-J-1640 and by Texas Advanced Research Program grant 003658–065.

show that under the prescribed access conditions the same overall effect is achieved by the program. The local copies are, of course, transparent to the user; they are employed merely for implementation of a shared variable by message communication. Therefore, the programming of tightly-coupled systems retains much of its succinctness while admitting a somewhat easier methodology for reasoning.

A problem in hardware design, called *cache-coherence*, has its roots in shared-variable programming. Suppose that several processes hold copies of a shared variable in their local caches. If processes write into their local copies then the local copies can become inconsistent; different processes may read different values of the shared variable if their copies are inconsistent. The traditional solutions to cache-coherence restrict the accesses to the caches at run-time in such a manner that inconsistencies are avoided; one typical solution is to lock all caches prior to a write by any process, and then broadcast the new value to all processes after completion of the write. Our approach in this paper provides an alternative to cache-coherence: If our proposed access conditions on shared-variables are met then the caches need not be consistent; caches are never locked; any process changing its cache transmits the change to all other processes asynchronously, and a process receiving a message about a change effects the change to its cache appropriately.

We had proposed a set of access conditions in (see Chapter 8 [1]); the present work simplifies and generalizes those conditions.

2 Access Conditions for Shared Variables

Consider an integer variable x shared between processes A, B. Process A only increases x by a fixed amount (say Δ) and process B only decreases x (by Δ) ensuring that its value remains nonnegative. We claim that A, B are loosely-coupled because the nature of access to the variable shared between them provides for (almost) asynchronous execution. To see this, imagine that there is a channel from A to B along which A transmits the amount by which it increments x; process B has a local copy of x, call it y. Whenever B recieves a message with value Δ, it increments y by Δ. All references to x in B are replaced by references to y. Initially $y = x$. It can be seen that $y \leq x$ is an invariant. Therefore, if B executes a step by examining y $(y \geq \Delta)$, it would have taken the same step by examining x $(x \geq y \geq \Delta)$. Furthermore, the value of y "tends" to catch up with the true value of x because of the messages transmitted by A to B; hence, the system will not be permanently blocked. Thus, shared variable x has been implemented by message communication.

In the sequel, we formalize these discussions and propose general conditions under which shared variables can be implemented by messages. Note that any shared variable can be implemented as a process; every read and write request is then treated as a message to this process. Even more elaborate schemes are required if concurrent accesses to more than one shared variable are desired (a process can then be viewed as a drinking philosopher [1, Chapter 13] and a shared-variable as a beverage. A solution to the drinking philosophers problem guarantees that accesses to shared variables are mutually exclusive, and yet different processes can simultaneously access different shared variables.) Our interest is in defining conditions that allow us to implement shared variables by message communications using a minimal amount of overhead.

Let x be a variable that is shared among a set of processes, $P_0 \ldots P_n$. Introduce variables $x_0, \ldots, x_n$ be such that x_i is written only by P_i, $0 \leq i \leq n$, and for some function f,

$$x = f(x_0, \ldots, x_n)$$

This requirement merely states that the value of x is a composite of all the changes made to it by the component processes. For instance, suppose x is an integer and the processes increase or decrease x by fixed amounts. Postulate variables $x_0, \ldots, x_n$ where x_i is the amount by which process i has changed x; then,

$$x = x_0 + \ldots + x_n$$

Note, however, that if the processes change x to a value that depends on x's current value (say, by executing $x := x^2$) then it may not be possible to express x as $f(x_0, \ldots, x_n)$, for any function f.

As another example, let x be a sequence shared between processes A and B, where A appends items to the end of x and B may remove the head item of x provided x is nonempty. (Note that x implements message communication between A, B. Our implementation of x will carry out exactly the same message communications.)

$$x = x_A \ominus x_B$$

where x_A is the sequence of messages sent by A
x_B is the sequence of messages received by B

$x_A \ominus x_B$ is the sequence remaining after removing the prefix x_B from x_A. Initially, x_A, x_B are both empty sequences.

Finally, consider an example where x is a boolean—initially false—that can be updated to true by any process in $P_0 \ldots P_n$. Then,

$$x = x_0 \lor x_1 \lor \ldots \lor x_n$$

where x_i is set to true by P_i whenever it sets x to true. Initially all x_i's are false.

Next, we impose two access conditions on $x_0, \ldots, x_n$.

(write) The value of x_j, $0 \leq j \leq n$, never "decreases," i.e., there is a partial order, $\sqsubseteq_j$, over the possible values of x_j such that any two successive values of x_j—a, b—satisfy

$$a \sqsubseteq_j b$$

The write condition is a statement of monotonicity—that each component x_j is updated only in a monotonic fashion (with respect to $\sqsubseteq_j$). The read condition, stated below, postulates that any value assigned to a local variable of a process remains valid even if another process changes the value of x.

Assume that a shared variable, x, is read only in a statement of the form

$$u := g(x) \quad \text{if} \quad b(x)$$

(that assigns $g(x)$ to u if $b(x)$ holds; if $\neg b(x)$ holds the process waits without changing u. See Chapter 2 of [1] for an explanation of the syntax.)

(read) Let

$$u := g(x) \qquad \text{if } b(x)$$

be a statement in process i, $i \neq j$. (The right side of the assignment may name variables other than x.) We write $g(x|x_j := s)$ for the expression obtained from $g(x)$ by first replacing x by $f(x_0, \ldots, x_n)$ and then replacing x_j by s; similarly, $b(x|x_j := s)$. We require

$$(s \sqsubseteq_j t) \wedge b(x|x_j := s) \Rightarrow b(x|x_j := t)$$

Additionally, if u is a local variable of process i then

$$(s \sqsubseteq_j t) \wedge b(x|x_j := s) \Rightarrow g(x|x_j := s) = g(x|x_j := t)$$

We give a few small examples to illustrate these conditions.

Example 1 (Counter)

Two processes, A, B, share an integer-valued variable x. The only statement in A naming x is

$$x := x + 1 \qquad \text{if } b \qquad \{b \text{ does not name } x\}$$

The only statement in B naming x is

$$x, u := x - u, u^2 \qquad \text{if } x \geq u \qquad \{u \text{ is a local variable of } B\}$$

To show that A, B are loosely-coupled, we introduce variables y and z, where

$$x = y - z$$

Initially, let $y = x$ and $z = 0$. Variable y is changed only in A whenever x is changed, i.e., the statement in A is changed to

$$x, y := x + 1, y + 1 \qquad \text{if } b$$

Variable z is changed only in B whenever z is changed, i.e., the statement in B is changed to

$$x, u, z := x - u, u^2, z + u \qquad \text{if } x \geq u$$

Both variables—y, z—are increased by A, B. Hence the write condition is met. To see that the read condition is met we have to show that the value assigned to u is unaffected by changes in y, i.e.,

$$s \leq t \wedge (y - z|y := s) \geq u \Rightarrow [(y - z|y := t) \geq u \wedge (u^2|y := s) = (u^2|y := t)]$$

This condition can be simplified to

$$s \leq t \wedge s - z \geq u \Rightarrow t - z \geq u \wedge u^2 = u^2$$

which clearly holds.

Note: Had the statement in process B been

$$x, u \ := \ x - u, x \qquad \text{if} \quad x \geq u$$

the read condition would not have been satisfied. We will show that x can be implemented as follows.

1. Replace x in the test "if $x \geq u$" by $y - z$ in process B.
2. Eliminate variable x from both programs, since it is an auxiliary variable.
3. Create a local variable y' that is a copy of y, in process B; variable y' is used in the place of y in process B.
4. Process A communicates the new value of y whenever y is changed, to process B.
5. Process B, upon receiving a value from process A, assigns the received value to y'. □

Example 2 (Bounded Communication)

Processes A, B share a variable x of type sequence whose length is at most 3. The statement in A that names x is of the form

{where variable v is local to Process A and $(x; v)$ is the concatenation of v to the end of x}

{send} $x \ := \ (x; v) \qquad \text{if} \quad |x| < 3$

The statement in B that names x is of the form

{receive} $x, u \ := \ \text{tail}(x), \text{head}(x) \qquad \text{if} \quad x \neq \text{empty}$

Introduce variables y, z, both unbounded sequences, where

$$x = y \ominus z$$

Variable y is the sequence of values appended by process A. Variable z is the sequence of values removed by process B, and $y \ominus z$, defined only if z is a prefix of y, is the suffix of y left over after removing z from y. Initially, let $y = x$ and z be empty. We modify the statement in A to

$$x, y \ := \ (x; v), (y; v) \qquad \text{if} \quad |x| < 3$$

The statement in B is modified to

$$x, u, z \ := \ \text{tail}(x), \text{head}(x), (z; \text{head}(x)) \qquad \text{if} \quad x \neq \text{empty}$$

The write condition is obviously met for y, z; here, $s \sqsubseteq t$ denotes that sequence s is a prefix of sequence t.

The read condition in A is

$$s \sqsubseteq t \ \wedge\ |(y \ominus z|z \ := \ s)| < 3 \ \Rightarrow\ |(y \ominus z|z \ := \ t)| < 3$$

The read condition in B is

$$s \sqsubseteq t \ \wedge\ (y \ominus z|y \ := \ s) \neq \text{empty} \Rightarrow$$
$$[\ (y \ominus z|y \ := \ t) \neq \text{empty} \wedge \text{head}(y \ominus z|y \ := \ s) = \text{head}(y \ominus z|y \ := \ t)\ ]$$

Both these conditions are met. Hence, x can be implemented by message communication, no big surprise! □

Example 3 (Multiprocess sharing)

We consider a boolean variable x shared between processes $P_0, \ldots, P_n$. Each process has a statement of the form

$$x \ := \ \text{true} \qquad \text{if} \quad b \wedge \neg x$$

where b is a local condition of the process. This is, typically, the skeleton of a program in which processes carry out a search, each process in a different part of a search space; any process that succeeds in its search sets x to true. A process discovering that x is true abandons the search.

Introducing boolean variables $x_0, \ldots, x_n$ where x_i is set to true by process i, we have

$$x = x_0 \ \vee\ \ldots\ \vee\ x_n$$

The read and write conditions can be verified for this program, using false $\sqsubseteq$ true.

3 Correctness

Let programs F, G share a variable x; we will show that no generality is lost by considering only two processes. Suppose

$$x = f(y, z)$$

where y is changed only in process F and z in process G. We replace x by $f(y, z)$ in both programs. Clearly, this step is correctness preserving. Next, we replace all occurrences of y in G by y', a local variable of G. We augment F to send the new values of y to G and for G to receive from A and appropriately update y'. We restrict our correctness arguments to showing that replacing y by y', as described above, preserves all the properties of the composite program consisting of F, G.

The formalism which I find convenient for stating and proving these properties is UNITY [1]. We have the following property in F, where Y is a free variable.

F1. $Y \sqsubseteq y$ is stable in F

which states that y can never "decrease" in F (with respect to $\sqsubseteq$). In G, no assignment is made to y; we assume that any statement naming y is of the form

$$u \ := \ g(y) \qquad \text{if} \quad b(y)$$

We have the read condition for such a statement (assuming that u is a local variable of G)

G1. For all s, t, $s \sqsubseteq t \ \wedge \ b(s) \ \Rightarrow \ b(t) \ \wedge \ g(s) = g(t)$

Construct the program G' from G by replacing y by y'. Next, we consider a program H to assign y to y', periodically, thus simulating message transmission. Program H does not affect any variable of F or G; in particular y is constant in H. We describe H by its properties.

H1. $Y \sqsubseteq y'$ stable in H
H2. $\langle \forall B$ ::
    $Y \sqsubseteq y$ stable in $B \ \wedge \ y'$ constant in B
    $\Rightarrow \ y' \sqsubseteq y$ invariant in $B \, [\!] \, H$
      $\wedge \ y = k \ \mapsto \ k \sqsubseteq y'$ in $B \, [\!] \, H$
    $\rangle$

Property H1 states that y' is never decreased in H. Property $H2$ states that if H is composed with any program B where y is never decreased and y' never changed then in $B \, [\!] \, H$, y' is at most y and y' eventually "exceeds" any previous value of y.

Theorem 1 shows that all safety properties of $F \, [\!] \, G$ are preserved in $F \, [\!] \, G' \, [\!] \, H$; theorem 2 shows that all progress properties, that are *leads-to*'s, are also preserved. In the following theorems p, q are arbitrary predicates not naming y'.

Theorem 1: p *unless* q in $F \, [\!] \, G$
  $\Rightarrow \ p$ *unless* q in $F \, [\!] \, G' \, [\!] \, H$

Theorem 2: $p \mapsto q$ in $F \, [\!] \, G$
  $\Rightarrow \ p \mapsto q$ in $F \, [\!] \, G' \, [\!] \, H$

The proofs use only the given properties F1, G1, H1 and H2.

References

1. Chandy, K. M., and J. Misra [1988]. *Parallel Program Design: A Foundation*, Reading, Massachusetts: Addison-Wesley, 1988.

Hierarchical Lazy Caching

Geoffrey Brown * Michael Merritt †

Abstract

This paper explores the application of lazy caching [1] to hierarchical cache architectures such as that suggested in [12]. Lazy caching enforces a weaker definition of consistency than commonly required. This results in an efficient protocol for hierarchical systems which has the performance of the more complex protocols which have been suggested, but is easier to implement. Furthermore, the weaker consistency definition used does not compromise correct program execution.

1 Introduction

In shared memory multiprocessors the latency for memory access grows logarithmically with the number of processors. Caching is a basic technique used to shield the processors from this increased latency by speeding access to frequently used data. Unfortunately, the consistency protocols required to maintain the illusion of a single shared memory result in contention at the caches. In large systems this contention can seriously degrade performance. One common proposal for extending the current generation of protocols to larger systems is to structure the memory system as a hierarchy of caches [6, 13, 3, 4, 10, 12]

In a hierarchical caching system, the invalidation traffic for shared data can be isolated to that portion of the system where the data is being shared. Provided that processes accessing shared data are localized, the remainder of the system is shielded from that invalidation traffic. However, implementing efficient consistency protocols for hierarchical caching systems can be difficult. In this paper we show that hierarchical lazy caching can provide high efficiency while maintaining ease of implementation.

Lazy caching supports a weaker consistency property than do most other protocols. However, this consistency property is still sufficient to produce correct behavior in asynchronous concurrent programs. The algorithm exploits this greater freedom to reduce synchronization between processors and local and remote caches. Specifically, writes to shared memory are buffered once when leaving the writing processor, and again coming into each local cache. Processing of these buffered operations continues in parallel with processor activity–processors are delayed only on reads of shared memory that are preceded by writes. Hierarchical lazy caching is an application of these techniques to an hierarchical architecture shown in Figure 1.

Two additional reasons that hierarchical caches are of interest are:

---

*School of Electrical Engineering, Cornell University.

Supported by NSF Grant CCR-9058180.

†AT&T Bell Laboratories.

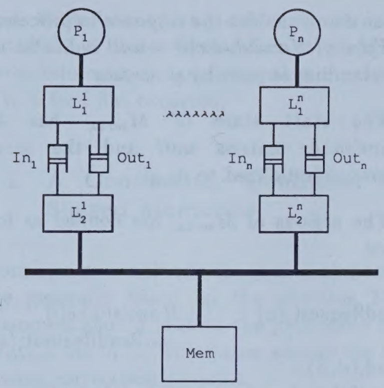


Figure 1: Hierarchical Multicache Systems

- Latency increases with memory size – for many applications it is desirable to use very large private caches. Unfortunately, the latency of these large caches is likely to be greater than for a smaller cache. One solution to this latency problem is to back a small fast cache with a larger slower cache.

- Chip area limits cache size – most of the current generation of microprocessors have on-chip caches (e.g. Motorola 68020). Because the size of these caches is limited by chip area, it is common to supplement them with larger secondary off chip caches.

The remainder of this paper is organized as follows. Section 2 describes the consistency criterion from [1]; Section 3 describes the hierarchical lazy caching algorithm and presents a proof that it satisfies the consistency criterion. Section 4 describes simulation experiments that suggest that the performance gains from hierarchical lazy caching may equal those of more complex approaches. At the same time, the more relaxed synchronization requirements of lazy caching may simplify hardware implementations. Such simplifications may be of particular significance in parallel processors incorporating microprocessor chips with on-board caches. This issue is discussed in a final section.

2 Memory Systems*

The problem of data consistency arises whenever systems allow multiple copies of a datum to exist; updating one of the copies results in an inconsistent system state which, if not properly handled, may result in erroneous system behavior. In multiprocessor systems the task of ensuring that cached copies of data are consistent falls upon the, so-called, consistency protocol. Traditional consistency protocols have required that all accessible copies of a datum be identical at all times[11]. While a system implementing such a protocol is certainly correct, the discipline required to enforce such a property may unnecessarily constrain concurrency in the system. This section formally defines a weaker consistency criterion for shared memory that is satisfied by the hierarchical lazy cache algorithm of Section 3. Discussion and motivation for this consistency criterion appears in [1].

Following [8, 7], a system is defined as a set of states and a set of actions which are multivalued functions from states to states. A system gives rise to a set of *executions*, which are finite or infinite sequences $s_0, \pi_1, s_1, \ldots$ of alternating states (the s_i's) and actions (the π_i's), where $s_i \in \pi_i(s_i - 1)$ and s_0 is an initial state. The actions of a system are partitioned into two classes: *external* actions and *internal* actions. The *behaviors* of a system are the sequences of external actions described by its executions; that is, the set of projections of the executions on the set of external actions. (If β is a behavior, and P is a set of actions, we denote by $\beta|P$ the subsequence of β consisting of the actions in P.)

An action is specified using the notation

$$\langle guard \rangle \rightarrow \langle command \rangle$$

where the *guard* is a boolean expression and the *command* is sequence of assignments. An action can only be executed from a state in which its guard is true. The guard of an action may be omitted if it is true in all states.

---

*Much of the material in this and the succeeding section is taken from [1].

This paper is concerned with the behavior of memory systems as viewed at the processor-memory interface. An n-processor shared memory is characterized by a particular set of external actions by which it interacts with the processors. In particular, such memory systems have four types of external actions:

$\mathsf{ReadRequest}_i(a)$
(P_i requests the value in address a.)
$\mathsf{ReadReturn}_i(d, a)$
(P_i receives data d from address a.)
$\mathsf{WriteRequest}_i(d, a)$
(P_i requests a write of d to address a.)
$\mathsf{WriteReturn}_i(d, a)$
(P_i is notified that a write is complete.)

These are called the *external memory actions.* Particular memory systems may have additional internal actions.

A conventional memory system, denoted M_{serial}, is defined by specifying its internal actions and state variables. In addition to the external actions above, M_{serial} has two internal actions, $\mathsf{Read}(d, a)$ and $\mathsf{Write}(d, a)$. These actions are the atomic events when the read and write operations actually occur. The states of M_{serial} consist of two arrays:

- $Mem[address]$, with entries of type $Data$, which records the current state of the memory, and
- $Handshake[1...n]$, which takes as values either external actions of M_{serial} or $null$.

The array $Handshake$ stores outstanding memory operations: $Handshake[i] = \mathsf{WriteRequest}(d, a)$ indicates an outstanding write request by processor i of value d to memory location a, and $Handshake[i] = \mathsf{WriteReturn}(d, a)$ indicates the write operation has occurred and enables the response to processor i. Similarly, $Handshake[i] = \mathsf{ReadRequest}(a)$ indicates an outstanding read request by processor i of memory location a, and $Handshake[i] = \mathsf{ReadReturn}(d, a)$ indicates the read operation has obtained the value d and enables the response to processor i. Finally, $Handshake[i] = null$ indicates no outstanding request by processor i.

The start state of M_{serial} has all $Handshake$ entries $null$ and the array $Memory$ initialized to d_{init}.

The actions of M_{serial} are defined as follows:

$$\begin{array}{lll} \mathsf{ReadRequest}_i(a) :: & & Handshake[i] := \mathsf{ReadRequest}_i(a) \end{array}$$

$$\begin{array}{lll} \mathsf{Read}_i(d, a) :: & & \\ Mem[a] = d \wedge Handshake[i] = \mathsf{ReadRequest}_i(a) & \rightarrow & Handshake[i] := \mathsf{ReadReturn}_i(d, a) \end{array}$$

$$\begin{array}{lll} \mathsf{ReadReturn}_i(d, a) :: & & \\ Handshake[i] = \mathsf{ReadReturn}_i(d, a) & \rightarrow & Handshake[i] := null \end{array}$$

$$\begin{array}{lll} \mathsf{WriteRequest}_i(d, a) :: & & Handshake[i] := \mathsf{WriteRequest}_i(d, a) \end{array}$$

$$\begin{array}{lll} \mathsf{Write}_i(d, a) :: & & \\ Handshake[i] = \mathsf{WriteRequest}_i(d, a) & \rightarrow & Mem[a] := d \\ & & Handshake[i] := \mathsf{WriteReturn}_i(d, a) \end{array}$$

$$\begin{array}{lll} \mathsf{WriteReturn}_i(d, a) :: & & \\ Handshake[i] = \mathsf{WriteReturn}_i(d, a) & \rightarrow & Handshake[i] := null \end{array}$$

$$i \in \{1, ..., n\},\ d \in data,\ a \in address$$

Thus, $\mathsf{ReadRequest}_i(a)$ can be executed from any state and has the effect of recording a request to read from address a, $\mathsf{Read}_i(d, a)$ can only be executed in a state where $Mem[a] = d$ and where processor i has requested data from address a, and $\mathsf{ReadReturn}_i(d, a)$ can only be executed in a state where $Handshake[i] = \mathsf{ReadReturn}_i(d, a)$, indicating that a read operation for i has occurred. Similarly, $\mathsf{WriteRequest}_i(d, a)$ can be executed from any state and has the effect of recording a write request, $\mathsf{WriteReturn}_i(d, a)$ can only be executed from a state in which $Handshake[i] = \mathsf{WriteRequest}_i(d, a)$ and has the effect of modifying Mem by setting the value of $Mem[a]$ to d and of setting $Handshake[i]$ to $\mathsf{WriteReturn}_i(d, a)$, and WriteReturn$_i(d,a)$ can only be executed in a state where $Handshake[i]$ = WriteReturn$_i(d,a)$, indicating that a write of d to a for i has occurred. †

2.1 A Consistency Condition for Shared Memories

Consistency conditions for shared memories are generally based on the intuition that memories should look to the processors like M_{serial}. As in [1], this paper adopts the following correctness criterion:

Definition 1 *A memory M is consistent only if, for each of its behaviors β, there exists a behavior β_s of M_{serial} such that for all processors $i, \beta|P_i = \beta_s|P_i$.*

Where $\beta|P_i$, for β a behavior, denotes the subsequence of β consisting of the actions of P_i, the actions with subscript i.

This condition requires that for each behavior of the cache memory system, there exist a behavior of the conventional shared memory such that, for each processor individually, the two behaviors are identical at the interface of that processor with the memory system. (This is similar to the serializability condition that most concurrency control algorithms impose on committed transactions.)

Motivation for this correctness condition rests heavily on the presumed architecture of the system, i.e. that processors communicate only through shared memory. If processors have an independent means of communication, or accurate measures of real time, it is possible for them to observe anomalies by comparing the relative times of different memory operations. See [1] for a more complete discussion.

3 The Hierarchical Lazy Cache Algorithm

The algorithm in this section combines the ideas of [1] with traditional write-through cache consistency algorithms [11] in a hierarchical cache scheme.

In addition to the central memory Mem, a pair of private caches L_1^i and L_2^i are associated with each processor i. The L_1^i arrays may become "out of date", but still be referenced by their associated processor. Like more traditional schemes, the L_2^i caches and the central memory track a consistent sequential history of the memory.

As indicated in Figure 1, this process is managed by a pair of queues, Out_i and In_i, associated with each cache. The Out_i queue buffers write operations by processor i, and the In_i queue buffers update operations to the cache L_1^i. Instead of updating the Mem array as in M_{serial}, action WriteReturn$_i(d,a)$ by processor i appends the pair (d,a) to Out_i. As a separate atomic MemoryWrite$_i$ action, the entry at the head of Out_i is stored in the array Mem and in the local L_2^i cache, is appended to In_i and is also appended to the In queues and inserted in the L_2 caches of those processors which currently hold a copy of the corresponding memory word. This effectively delays the atomic occurrence of the write until the appropriate entry reaches the head of Out_i. Occurrences of the CacheUpdate$_i$ action use the entries in In_i to update the contents of the cache L_1^i.

Suppose that processor i writes value d to memory location a and reads the same location, with no intervening operations by other processors. The value returned must be the value written. To assure that this and similar constraints are satisfied, the algorithm ensures that the state of the cache L_1^i used by a read operation reflects values in Mem that are at least as current as the most recent write by the same processor. Accordingly, a read operation cannot follow a write by the same processor until the entry for that write moves out of Out_i and the associated update moves through In_i queue into L_1^i.

---

† The external actions of M_{shared} model a handshake between each processor and the shared memory; the ReadRequest$_i$ and WriteRequest$_i$ actions model inputs from processor i to the memory, and the ReadReturn$_i$ and WriteReturn$_i$ actions are outputs from the memory to processor i. If a processor violates the handshake (by issuing a new request before a reply has been output) the previous request is overwritten. If the previous request was for a write operation, the write may or may not have occurred, depending on whether the enabled internal Write action had fired. A less constraining specification might permit other behavior in the presence of such an erroneous environment.

To accomplish this, entries in the In queues that record writes by processor P_i are tagged with a special character ("*"). Responses to reads (ReadReturn$_i$'s) are not allowed to proceed unless the Out_i queue is empty and there are no tagged entries in In_i.

Two types of internal actions are enabled in every state: MemoryRead$_i$ actions add any $(data, address)$ pair from Mem to In_i and L_2^i, and CacheInvalidate$_i$ actions remove entries from L_1^i and L_2^i. (Both maintain the invariant that L_1^i, properly updated with the entries in In_i, is a subset of L_2^i.) Allowing these actions nondeterministically from any state avoids a need to specify the memory-management strategy that might be chosen to enhance efficiency in any particular system. Our proof of correctness applies *a fortiori* to such strategies by restriction of nondeterminism.

Such an algorithm may be more efficiently implementable than more traditional algorithms because processors are only delayed during read and write operations in the case that a read follows a write or does not find the needed value in the cache. Furthermore, atomic synchronization occurs at the events updating the L_2^i caches and appending entries to the In queues, not at the L_1^i caches. Processors may continue to access these local, fast caches in parallel with this activity.

3.1 A Precise Description of the Algorithm

The hierarchical cache memory system $M_{hier-cache}$ has the same external actions as M_{serial}, but has more complex data structures and additional internal actions. The data structures include Mem of M_{serial} and the following:

- n pairs of unbounded fifo queues, or lists, In_i and Out_i, the entries in which are either $(data, address)$ or $(data, address, *)$ tuples. These queues have the following operations:
  - $append(queue, item)$, adds $item$ as the last entry in $queue$.
  - $head(queue)$, returns the first entry in $queue$.
  - $tail(queue)$, returns the result of removing $head(queue)$ from $queue$.
  - $\{\}$ denotes the empty queue.
  - $queue[i]$ denotes the ith element of $queue$ where $queue[0] = head(queue)$
- n partial functions $L_1^i : Address \times Data$.
- n partial functions $L_2^i : Address \times Data$. These functions may be updated via two operations:
  - $update(L_1^i, d, a)$ returns the partial function which maps address a to value d, but is otherwise identical to L_1^i. Similarly, $update(L_1^i, In_i)$ is the partial function obtained by successively *updating* L_1^i with the $(data, address)$ pairs from entries in In_i.
  - $restrict(L_1^i, L_2^i)$, which nondeterministically chooses any pair of partial functions (l_1^i, l_2^i), such that $l_1^i \subseteq L_1^i$, $l_2^i \subseteq L_2^i$, and $update(l_1^i, In_i) \subseteq l_2^i$.

The initial states of $M_{hier-cache}$ are those in which all queues are empty, the array Mem is initialized to d_{init}, and the $L_1^i \subseteq L_2^i \subseteq Mem, \forall i \in \{1, ..., n\}$.

In addition to the external actions of M_{serial}, the actions of $M_{hier-cache}$ include the internal actions discussed above: MemoryWrite$_i(d, a)$, MemoryRead$_i(d, a)$, CacheUpdate$_i(d, a)$, CacheInvalidate$_i$.
The actions of $M_{hier-cache}$ are specified as follows:

ReadRequest$_i(a)$:: $Request[i] := a$

ReadReturn$_i(d, a)$::
$Request[i] = a$ $\rightarrow$ $Request[i] := null$
$\wedge L_1^i[a] = d$
$\wedge Out_i = \{\}$
$\wedge$ (There are no starred entries in In_i.)

WriteRequest$_i(d, a)$:: $Request[i] := (d, a)$

WriteReturn$_i(d, a)$::
$Request[i] = (d, a)$ $\rightarrow$ $Request[i] := null;$
$Out_i := append(Out_i, (d, a))$

CacheUpdate$_i(d, a)$::
$head(In_i)$ is either (d, a) or $(d, a, *)$. $\rightarrow$ $In_i := tail(In_i);$
$L_1^i := update(L_1^i, d, a)$

CacheInvalidate$_i$:: $(L_1^i, L_2^i) := restrict(L_1^i, L_2^i)$

MemoryWrite$_i(d, a)$::
$head(Out_i) = (d, a)$ $\rightarrow$ $Mem[a] := d;$
$Out_i := tail(Out_i);$
$In_i := append(In_i, (d, a, *));$
$L_2^i := update(L_2^i, d, a);$
$(\forall k : (k \neq i)$
$\wedge (a \in domain(L_2^k)) :$
$In_k := append(In_k, (d, a));$
$L_2^k := update(L_2^k, d, a))$

MemoryRead$_i(d, a)$::
$Mem[a] = d$ $\rightarrow$ $In_i := append(In_i, (d, a))$
$\vee L_2^i[a] = d$
$L_2^i := update(L_2^i, d, a))$

$$i \in \{1, ..., n\},\ a \in addresses,\ d \in data$$

The correctness proof makes use of the following simple property of $M_{hier-cache}$:

Lemma 1 *Every reachable state of $M_{hier-cache}$ satisfies the following properties:*

- $(\forall i :: update(L_1^i, In_i) \subseteq L_2^i)$
- $(\forall i :: L_2^i \subseteq Mem)$

3.2 A Proof of Correctness of $M_{hier-cache}$

The hierarchical cache algorithm is essentially an elaboration of the lazy cache algorithm of [1], which was formulated as the automaton M_{cache}. The proof of correctness makes that relationship explicit, by mapping states and transitions of $M_{hier-cache}$ to states and transitions of M_{cache}. The properties of this mapping imply that behaviors of $M_{hier-cache}$ are also behaviors of M_{cache}; the correctness of $M_{hier-cache}$ follows as a corollary to that of M_{cache}, the main theorem of [1].

The states of M_{cache} are obtained from those of $M_{hier-cache}$ by deleting the second level caches L_2^i. The action names are identical, and the actions themselves are identical except for CacheInvalidate, MemoryWrite and MemoryRead, which are defined below. ‡ (Here, $restrict(L_1^i)$ returns an arbitrary subset of L_1^i.)

CacheInvalidate$_i$:: $L_1^i := restrict(L_1^i)$

MemoryWrite$_i(d, a)$::
$head(Out_i) = (d, a)$ $\rightarrow$ $Mem[a] := d;$
$Out_i := tail(Out_i);$
$In_i := append(In_i, (d, a, *));$
$(\forall k : k \neq i :$
$In_k := append(In_k, (d, a)));$

MemoryRead$_i(d, a)$::
$Mem[a] = d$ $\rightarrow$ $In_i := append(In_i, (d, a))$

$$i \in \{1, ..., n\},\ a \in addresses,\ d \in data$$

Projecting the second level caches (the L_2^i) from states of $M_{hier-cache}$ produces a state of M_{cache}. The actions of $M_{hier-cache}$ correspond almost exactly to those of M_{cache}. The correspondences for the external actions ReadRequest, ReadReturn, WriteRequest and WriteReturn and for CacheUpdate and

---

‡In [1], the notation C_i was used in place of L_1^i.

CacheInvalidate are immediate. The correspondence for **MemoryRead** follows from Lemma 1.

The only complication is the filtering that takes place in the **MemoryWrite** action in $M_{hier-cache}$, where, again by Lemma 1, the update request is propogated only to the in-queues at processors which have any copy of the relevant memory word.

The **MemoryWrite** action of M_{cache} instead propogates the update requests to *all* in-queues. If there is no copy of a memory word with address a in In_i or in L_1^i, then the consequence of entering (d, a) into In_i, propogating it into the cache L_1^i and removing it via a **CacheInvalidate** action is the same as not entering it into In_i at all. A straightforward though tedious history variable construction formalizes this intuition and proves:

Theorem 1 *Every behavior of* $M_{hier-cache}$ *is a behavior of* M_{cache}.

4 Simulations

This section discusses the results of simulation experiments used to analyze the performance of the hierarchical lazy caching algorithm. It compares this algorithm with a two level cache system using pure write-through (no invalidation or write buffering) and with a single level cache system using a variant of the Dragon protocol as described by Archibald and Baer [2]; those experiments have indicated that it is one of the best consistency protocols in terms of performance. The experiments described here indicate that either two level cache system performs better than a single level cache system when the cost of a level 1 cache cycle is less than a level 2 cache cycle or a bus cycle. Furthermore, lazy caching provides an additional performance advantage over a pure write-through protocol in a two level cache system.

The simulation models were developed following the approach used in [9]. The principal difference in the models is that we assume separate data and instruction caches.

Operation	Bus Time
Instruction execution	0
In-cache read or write	0
Clean miss	7
Dirty miss	11
Write through	1

Table 1: Bus time for hardware operations

The simulations assume that data is statically partitioned as shared or nonshared (i.e. by a compiler). Thus bus operations to preserve consistency are necessary only for shared data; dirty copies of non-shared data are written back to main memory only when the corresponding cache line is needed due to a miss. Table 1 lists the number of bus cycles necessary for the various memory operations. The cost of memory operations is based upon a block size of four words. A clean miss needs 7 bus cycles, 1 to send the address, 2 for memory access, and 4 to get the block of data. In addition, the processor must access either the instruction or data cache at least once for every operation — twice for those operations resulting in a cache miss. The length of bus cycles and level 2 cache cycles are twice those of level 1 cache cycles. Instruction caches are assumed to be as fast as level 1 caches.

The workload model is also based upon that in [9] (similar workload parameters were used in [2]). This section presents only the results for "medium" values of the workload parameters as summarized in Table 2. A few of these parameters require further explanation. The miss rate for data refers to the overall data miss rate for the cache hierarchy; in the case of Dragon, this is the data miss rate for the single cache. For the hierarchical systems, a miss rate of 10% for the level 1 caches and a miss rate of 14% for the remaining accesses to level 2 caches was assumed. Previous simulation results [10] indicate that these rates correspond reasonably to an 8K level 1 cache backed by a 512K level 2 cache. The expected speed advantage of the smaller level 1 cache is reflected in the difference in cycle times. In modelling Dragon

probability an instruction is a load or store	0.3
miss rate for data	0.014
miss rate for instructions	0.0022
probability a miss replaces a dirty block	0.20
probability a load or store is a store	0.25
probability a load or store refers to shared data	0.25
number of caches containing a shared block on write broadcast	1.0
Dragon Only	
probability that a shared block in one cache is present in another	0.79

Table 2: Workload parameters

we assumed a large cache with level 2 cycle times [§].

In Dragon, the protocol keeps track of which shared data is actually in use by more than one processor through bus snooping and a special bus interface. This knowledge is imperfect and there may be times at which a processor believes that data is shared when it is not. This effect is modelled with a parameter which reflects the probability that a processor performing a write to shared data believes that another copy of the datum is in use.

In Figure 2 presents the results of our simulation for the parameters given. Notice that our lazy cache scheme has a 2%-4% performance advantage over a hierarchical system using pure writethrough. This advantage was maintained for simulation experiments with both higher and lower probabilities an instruction is a load or store, and corresponding higher and lower probabilities a load or store refers to shared data. (These correspond to "low" and "high" rates in [9]). Previous simulations for hierarchical caches in uniprocessor systems [10] indicate that using a write-back strategy in the level 1 caches instead of write-through leads to at most a 5% performance improvement. The disadvantage to using write-back is that maintaining consistency and the multi-level inclusion property is extremely difficult [10, 3]. Thus, using a lazy cache protocol for the level 1 caches provides most of the performance advantage of write-through without the concomitant implementation problems.

The results for Dragon indicate that hierarchical caching provides a performance benefit over single level caching even when write-through is used for cache coherence in the hierarchical system. At higher probabilities of load and store instructions, higher probabilites these reference shared data, and with larger numbers of processors, this performance advantage disappears due to increased bus contention. The lazy cache algorithm could also be used with a better level 2 consistency protocol such as Dragon. With this combination, one can expect further performance advantages.

5 Lazy Caching and Microprocessors

A major advantage of using the lazy cache algorithm for systems with on-chip caches is to help overcome the I/O bandwidth limitations of microprocessors. The additional buffering admitted by lazy caching facilitates the use of a dual-directory scheme with one directory off-chip. Only the line numbers of those lines which need to be invalidated have to be passed on chip. The weaker synchronization constraints allow the processor to continue execution even while invalidation requests are being handled off-chip. This is illustrated in Figure 3 where a duplicate cache directory is used off chip to limit the flow of invalidation requests through the chip's I/O pins.

In general, a hierarchical caching system renders the consistency problem more complex. A common proposal is to use a write-through policy for the L^1 cache and copy-back for the higher levels of the hierarchy. This approach ensures that the contents of

---

[§] Simulation results for the alternative, a small fast cache, indicate much worse performance

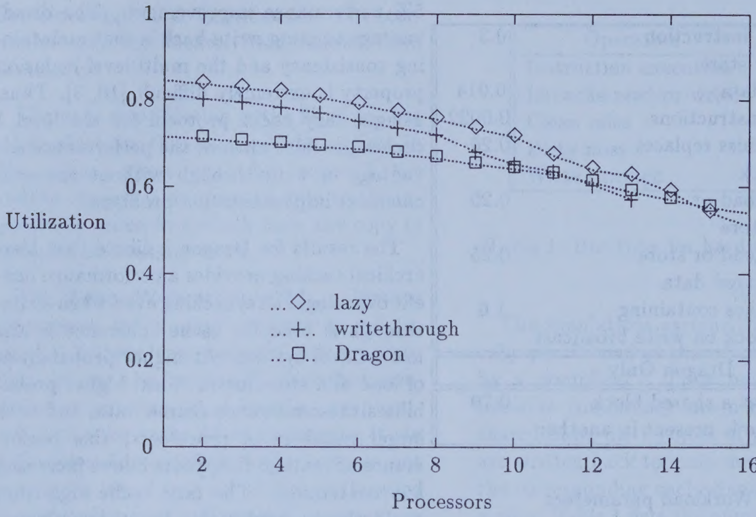


Figure 2: Performance of Cache Coherence Schemes

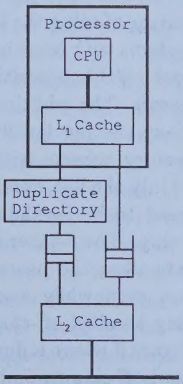


Figure 3: Lazy Caching for Microprocessors

L^1 are a subset of the contents of L^2 and simplifies the consistency algorithm. In [12] it is suggested that there are several problems with this approach. The primary complaint is that updating the first level cache requires updating the second level cache as well. Although the authors suggest that the write latency can be reduced by buffering and give substantiating performance data, they state that write buffers complicate the concurrency protocol. As we have shown in the previous section, this is true only if the traditional consistency constraint is used. The use of lazy caching has the potential to greatly simplify the implementation of the first-level write-through protocol.

References

[1] Yehuda Afek, Geoffrey M. Brown, and Michael Merritt. Lazy caching. SPAA '89 (Submitted to ACM TOPLAS.)

[2] James Archibald and Jean-Loup Baer. Cache coherence protocols: Evaluation using a multiprocessor simulation model. *ACM Transactions on Computer Systems*, 4(4):273–298, November 1986.

[3] Jean-Loup Baer and Wen-Hann Wang. Architectural choices for multilevel cache hierarchies. In *Proceedings of the 1987 International Conference on Parallel Processors*, pages 258–261, 1987.

[4] Jean-Loup Baer and Wen-Hann Wang. Multilevel cache hierarchies: Organizations, protocols, and performance. *Journal of Parallel and Distributed Computing*, 6:451–476, 1989.

[5] Susan J. Eggers and Randy H. Katz. A characterization of sharing in parallel programs and its application to coherency protocol evaluation. In *Proceedings of the 15th International Symposium on Computer Architecture*, pages 373–382, June 1988.

[6] Akira Hattori, Minoru Koshino, and Shigemi Kamimoto. Three-level hierarchical storage system for FACOM M-380/382. In *IFIP Information Processing 83*, pages 693–697, 1983.

[7] Nancy A. Lynch. I/O automata: a model for discrete event systems. Technical Report MIT/LCS/TR-351, MIT, March 1988. Also in 22nd Annual conference on Information Science and Systems, Princeton NJ, March 1988.

[8] Nancy A. Lynch and M. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the 6th ACM Symposium on Principles of Distributed Computation*, pages 137–151, August 1987. Expanded version available as Technical Report MIT/LCS/TR-387, April 1987.

[9] Susan Owicki and Anant Agarwal. Evaluating the performance of software cache coherence. In *Third International Conference on Architectural Support Programming Languages and Operating Systems*, pages 230–242, April 1989.

[10] Robert T. Short and Henry M. Levy. A simulation study of two-level caches. In *Proceedings of the 15th International Symposium on Computer Architecture*, pages 81–88, June 1988.

[11] A. J. Smith. Cache memories. *Computing Surveys*, 14:473–540, 1982.

[12] Wen-Hann Wang, Jean-Loup Baer, and Henry M. Levy. Organization and performance of a two-level virtual-real cache hierarchy. In *Proceedings of the 16th International Symposium on Computer Architecture*, pages 140–148, June 1989.

[13] Andrew W. Wilson Jr. Hierarchical cache/bus architecture for shared memory multiprocessors. In *Proceedings of the 14th International Symposium on Computer Architecture*, pages 244–252, June 1987.

COMPUTING WITH TIME
(TEMPORAL DIMENSIONS IN DISTRIBUTED COMPUTING)

Nicola Santoro

Center for Parallel and Distributed Computing (PARADISE)
Carleton University, Ottawa, Canada
and
Istituto di Scienze dell'Informazione
Universitá degli Studi di Bari, Bari, Italy

1. INTRODUCTION

Time is an human artifact superimposed on nature in our attempt to quantify its qualitative processes. This quantification has the immediate effect of discretizing the perceived continuum and enabling its measurement. It also has the deeper consequence of creating a "reality" which is *artificial*, in that it is the result of an artifact, and *abstract*, in that it is a human mind's creation, but which we can nevertheless observe, explore, describe, and analyze. The validity of an artificial and abstract reality, or *model*, rests ultimately with its usefulness in our interaction with the natural and concrete reality we are bound to operate in. There is little doubt on the *usefulness* of the temporal reality. (There are however significant doubts on the *value* of such a use.) A simple application of the reflexive capability of the mental processes leads to the definition of artificial and abstract realities of artificial and abstract realities; that is, to *models of models*.

Distributed computing is a model of an environment, a *message-passing system,* which is itself a model of many artificial systems (e.g., distributed systems, communication networks, systolic architectures, etc.); it provides a language to describe its components, its behaviour, its properties; furthermore, it includes the tools for the analysis and the measurement of such environment. In the language of distributed computing, a message-passing system is a collection of entities which communicate by transmitting messages. A *message* is a finite (and, possibly, bounded) sequence of bits. Each *entity* has local processing and storage capabilities, and has access to a local clock. The behaviour of the entities is *finite-state* and *event-driven*; that is, each entity at any time is in a particular system state (from a finite set of states) and, when a predefined external event occurs (e.g., a message is received, the local clock is increased by one unit, etc.), it will serially perform some operations whose nature depends on the current state and on the occurred event. The operations that can be performed are local computations, transmission of messages, and changes of state. Thus, the *behaviour* of an entity is a set of *rules* of the form State*x*Event->Action, where State is a system state, Event is one of a predefined set of external events, and Action is an indivisible sequence of operations. The *operational behaviour* of the system, also called a *distributed algorithm* or *protocol*, is the collection of the behaviour of all its entities; it is said to be *symmetric* if all entities have the same behaviour.

Entities communicate by sending messages. A binary relation, called out-*neighbour*, on the entities defines for each entity x the subset of the other entities, called out-*neighborhood*, to which x can send a message; analogous is the definition of in-*neighbourhood* of an entity. If, for each entity, its in-neighborhood coincides with its out-neighborhood, we will use the terms neighbour and neighbourhood. The couple G=(*V*,*E*) where *V* is the set of entities and *E* is the out-neighbour relation defines a graph G which describes the *communication topology* of the system. Hence, graph-theoretic concepts and terminology (e.g., nodes, edges,diameter, etc.) can be used to describe distributed algorithms and analyze their performance. Messages received at an entity are processed there in the order they arrive; if more than one message arrives at the same entity at the same time, they will be processed in arbitrary order. Both entities and transmissions are prone to failures.

The basic model is based on only two simple axioms: *Local Orientation* and *Finite Delays*. The Local Orientation axiom simply states that every entity can distinguish between its (in- and out-) neighbours, and can detect from which in-neighbour a received message was sent. As a consequence of the Local Orientation axiom, it can be assumed that each entity x has a distinct label associated to each out-edge (i.e., edge connecting x to an out-neighbour) and in-edge (i.e., edge connecting an in-neighbour to x). The Finite Delays axiom states that, in absence of failure, a message sent to an out-neighbour is eventually received there in its integrity. Note that this axiom does not imply the existence of any bound on transmission delays; it only states that, in absence of failure, a message will arrive after a finite delay without corruption.

With respect to *time*, the basic model does not make any assumption on the local clocks nor (except for the fact that it is finite) on transmission time (which include both processing and queuing delays). For these reason, the systems described by the basic model are commonly referred to as *fully asynchronous* ones, and represent one extreme case of the spectrum of message-passing systems with respect to time. On the other extreme are *fully synchronous* systems; that is systems defined by the following two restrictions: *Synchronized Clocks* and *Bounded Transmission Delays*. The Synchronized Clocks axiom states that all local clocks 'tick' simultaneously (although they might not sign the same time); the Bounded Transmission Delays axiom states that there exists a known upper bound ∂ on the transmission (including queueing and processing delays) of a message. Since ∂ is known a priori to all entities and all local clocks tick simultaneously, the unit of time can be redefined so that, if a processor sends a message at local time t to a neighbour, the message is received and processed there at time t+1 (sender's time). To avoid paradoxical situations, it is assumed that at any time step only one message can be send to the same neighbour. Other cases lying between these two extremes can obviously be defined; e.g., *partial synchrony* and *partial asynchrony* [4,15,16].

The difference between these two extreme cases, fully synchronous and fully asynchronous systems, is significant from a complexity viewpoint. For example, the *election* problem when G is a ring of size n requires $\Omega(n \log n)$ messages in the fully asynchronous case [2,6,13], but can be solved in O(n) messages even when n is not known [6,7,16] in the fully synchronous case. The difference is even more evident when one considers that the *consensus* problem is unsolvable in the asynchronous case when just one entity may be faulty [5], while it can be achieved in the synchronous case even if almost one third of the entities are faulty.

Surprisingly, very little is known about computing in fully synchronous systems. Most of the research in distributed computing has focused on the study of the asynchronous case; in particular, with respect to the temporal dimension, the work has focused on the implementation of *virtual* and *logical clocks* (e.g., see [9,11]), and on the construction of *synchronizers* (e.g., see [1]) which provide a notion of "step time" for computations in the system. Both logical and virtual clocks provide a mechanism for ensuring a consistent temporal ordering of events in an asynchronous system. Synchronizers are protocols which can be superimposed on a distributed computation; their execution creates the equivalent of a global clock tick and thus allows a synchronous protocol (the underlying computation) to be executable in an asynchronous system.

The aim of this paper is to introduce some of the issues, properties and basic results regarding fully synchronous systems. Using some known as well new results, we will describe some of the unique aspects of computing with time.

2. COMPUTING WITH TIME

2.1 Complexity Measures

The communication complexity of a distributed algorithm is evaluated with respect to two basic parameters: transmission and time. In the fully asynchronous case, transmission complexity is typically measured in terms of total number of messages transmitted during a computation; to obviate to the fact that messages might have different length, sometimes the total number of *bits* transmitted during a computation is used as the measure of the transmission complexity of that computation. The time complexity is defined as the total delay from the time the first processor starts the computation to the time the last processor terminates its participation in the computation; in asynchronous systems, there is no *a priori* relation between transmission delays and "clock tick" nor between different processor clocks; hence, different interpretation of "time" may be possible, e.g. "ideal" time, length of longest sequence of casually related messages in all possible executions of the algorithm, etc. The definition of time complexity needs to be refined in the case of fully synchronous systems.

There is an underlying notion of "real" time in terms of which we express the distance Δ_S between two consecutive clock ticks in a fully synchronous system S, as well as the time complexity of a computation in S. By definition, Δ_S must be such that any message sent at a clock tick **t** arrives and is processed at its destination at clock tick **t**+1.Since, in general, the time transmission delays ∂_S in S for a message is a monotonically non-decreasing function of the length of that message, then $\Delta_S \geq \partial_S(m)$ where m is the maximum message length (expressed in bits). The *clock complexity* C_S of a computation in S is defined as the as the total number of clock ticks from the time the first processor starts the computation to the time the last processor terminates its participation in the computation. The *time complexity* T_S is thus just

$T_S=C_S\,\Delta_S$. From this definition it is clear that Δ_S is an important complexity measure.

Observe that, if ∂_S is monotonically increasing, the maximum message size m must be bounded: otherwise Δ_S (and, thus, the time complexity) would be unbounded. This implies that, if a computation requires the transmission of M>m bits, it is actually requiring the transmission of [M/m] messages, where [x] denotes the ceiling of x. Furthermore, since at most one message can be sent to a neighbour at a given clock tick, this implies that the transmission of M>m bits requires [M/m] clock ticks or, equivalently, [M/m] Δ_S time.

2.2 Basic Results

Among the many examples and results indicating the unique characteristics of fully synchronous systems, the best known and (perhaps) the most powerful is the one expressed by the following "folk" theorem:

Property Any finite sequence of bits can be communicated in S transmitting two messages, regardless of the message size.

Proof. Let π be the sequence of bits, and let x and y be the transmitting entity and its receiving neighbour, respectively. Consider the following protocol for x: 1. send a message; 2. wait Π clock ticks; and 3. send another message, where Π is the integer whose binary representation is π. The protocol for y is the following: 1. upon receiving the first message, set count to zero; 2. at each clock tick, if no message is received increase count by one, otherwise stop. Obviously, when y stops, count=Π. []

The above property is a striking example of the difference that computing with time makes: since the message size is irrelevant and since the string π is finite but arbitrary, the property states that any finite amount of information (e.g., the Encyclopaedia Britannica) can be communicated by transmitting just two bits. The property, as stated, is incomplete from a complexity point of view. In fact, in a synchronous system, time and transmission complexities are intrinsically related to a degree nonexistent in asynchronous systems. In the example above, a constant bit complexity is achieved at the cost of a time complexity which is *exponential* in the length of the sequence of bits to be communicated, as stated by the following reformulation of the above property:

Lemma 1 Any finite sequence of bits π can be communicated in S transmitting two bits in time $2^{|\pi|-2}\,\Delta_S$.

The constant -2 in the exponent follows from the fact that the two bits transmitted can be used to convey information about π. Another (less known) property along the same lines is the following

Lemma 2 Any finite sequence of bits π can be communicated in S transmitting three bits in time $2^{|\pi|**(1/2)}\,\Delta_S$.

These lemmas provide simple tools to transform any asynchronous algorithm into a synchronous one, as well as a method to derive directly the complexity of the resulting algorithm. In the following the time complexity of an asynchronous algorithm is defined as the length of the longest sequence of casually ordered messages over all possible execution of that algorithm.

Lemma 3 (Transformation Lemma)
Let A be an asynchronous algorithm which requires the transmission of M_A messages of size at most w in time T_A to perform a computation. Then there exists a synchronous algorithm B which performs the same computation in S transmitting 3 M_A bits with a time complexity of at most $2^{w**(1/2)} T_A \Delta_S$.

Proof. Implement the transmission of each message in A using three bits as described above. The result then follows from Lemma 2.[]

This simple transformation mechanism might appear to yield inefficient solutions for the synchronous case. To dispel this appearance, consider the following classical problem.

Example. Election in a Synchronous Ring of Unknown Size.

Let G be a ring of size n, which is however unknown to the entities; and let each entity have a distinct value. The Election Problem is the problem of moving from an initial system configuration where every entity is in the same system state, to a configuration in which one entity is in a predefined state (say, *elected*) and all others are in a different predefined state (say, *defeated*); note that there is no a priori restriction on which entity should become *elected*. There are several asynchronous election algorithms for rings of unknown size which require the transmission of O(n log n) messages of size O(log **i**) and time O(n log n), where **i** is the largest entity value. By the transformation lemma it follows that it is possible to elect a leader in a ring of unknown size with O(n log n) bits and $O(\mathbf{i}^{1/2}$ n logn) time. This result must be compared with the bounds of the election algorithm by [6] and [16] specifically designed for synchronous systems: O(n log **i**) bits and $O(2^{\mathbf{i}}$ n) time. (In both cases we are assuming for simplicity that Δ_S=O(1).) In other words, the transformation lemma yields bounds which are order of magnitude better than those previously obtained by specifically designed algorithms.

The transformation lemma gives a basis of comparison for measuring the efficiency of synchronous solutions to problems for which there already exist asynchronous solutions. To improve on the bounds obtained by the use of the transformation lemma, it is necessary to more explicitly and cleverly exploit the availability of "time" as a computational tool. Some techniques which achieve this goal for some specific problems are described in the next section.

3. BASIC TECHNIQUES

The main goal is to exploit the availability of time as an explicit computational tool in synchronous systems. In these section, some techniques which make an explicit use of time are briefly discussed. An underlying aim is to avoid the transmission of unbounded messages; in particular, if the input values are drawn from some unbounded universe (e.g., positive integers) and the goal of the computation is the evaluation of a function of the input values, then the messages cannot contain such values. For example, the "trick" described in the previous section, and on which the Transformation Lemma is based, is an instance of a simple and direct way of exploiting time by *counting* it; in this case, the actual value is communicated but not transmitted.

3.1 Waiting

Waiting is a technique which uses time not to transmit a value (as in counting), but rather to ensure that a desired condition is verified. The simplest way of illustrating the technique is by means of an example.

Example. Minimum Finding in Ring of Known Size

Let G be a unidirectional ring of size n (known to all entities), and let each entity have a value (not necessarily unique). The Minimum Finding Problem is the problem of moving from an initial configuration where all entities are in the same system state, to a configuration where every processors whose associated value is the minimum of all the values is in a predefined state (say, *minimum*) and all others are in a different predefined state (say, *large*). Assume for the moment that all entities start the execution of the algorithm simultaneously. Consider now the following protocol described in terms of a generic entity x, where where $\mathbf{i}_x$ is the input value for x, and w=n.

Waiting Protocol

Wait $f(\mathbf{i}_x,w)$ unit of times; if, while waiting, a message is received, forward it along the ring and become *large*; otherwise, send a message and become *minimum*..

An appropriate choice of f guarantees that the Waiting Protocol correctly solves the minimum finding problem, under the assumption of simultaneous initiation. For example, let $f(a,b)= a\ b$. Then $\mathbf{i}_x\ n+d(x,y) < \mathbf{i}_y\ n$ for every y with $\mathbf{i}_x<\mathbf{i}_y$, where d(x,y) denotes the distance from x to y in G. In other words, every entity x with smallest value will finish waiting before the entities with larger value, become *minimum*, and send a message; furthermore, any such message will reach an entity with larger value while it is still waiting transforming it into *large*. The assumption on simultaneous initiation can be easily removed by executing a "wake up" routine (which will bound the difference between the initiation times of any two entities by n-1) before the Waiting Protocol, and choosing an appropriate f (e.g., $f(a,b)=a(b+1)$) [14]. It is interesting to note that the protocol solves the problem even if n is not known exactly but only within a constant factor c (i.e., $w \geq n/c$), using an appropriate f (e.g., $f(a,b)=a\ (b+c)$). In any case, the complexity is O(n) bits and O(n **i**) time.

Example. Minimum-Finding and Election in Networks of Known Size

The Waiting Protocol can be easily extended to work with any strongly connected G (not just rings); in this case, (exact or approximate) knowledge of the diameter d(G) can be substituted for the knowledge of n; the resulting complexity is O(e) bits and O(**i** d(G)) time, where e is the number of edges in G.

Consider again the Election problem mentioned in the previous section. This problem is solvable (i.e., a deterministic solution protocol exists) only if the input values are distinct. In this case, the Waiting Protocol solves the Election problem if d(G) is known within a constant factor: there will be only one *minimum* entity while all others are *large*. Furthermore, with an additional O(e) bits, a spanning-tree of G can be also be constructed [14].

3.2 Counting

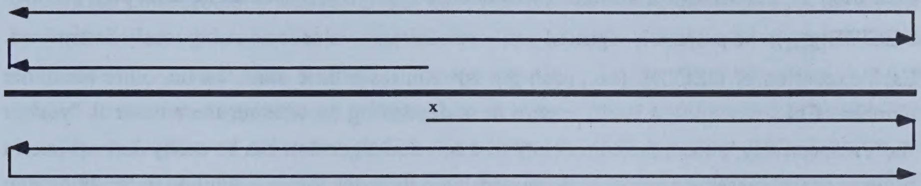
The property used in Lemmas 1 and 2 in the previous section, and on which the transformation lemma is based, is an instance of a simple and direct way of exploiting time by *counting* it; in this case, the actual value is communicated but not transmitted. In the following, we will give different examples of how counting can be used to efficiently solve problems in synchronous systems. Let G be a undirected tree of size n, and let all entities be in a situation of *topological ignorance* (i.e., they are aware only of their own neighbours).

Example. Eccentricity Determination

The eccentricity r(x) of an entity x is the maximum distance from x to all other entities in G. For simplicity assume that G is a simple path. To determine r(x), it is sufficient for x to send a message in both directions (just one if x is an endpoint of the path); along each direction, the message will be forwarded by each entity until it reaches an endpoint; the message will then be sent backwards along the path until it reaches x. The number of clock ticks from the time x sent the message to the time the message is received back clearly determines the maximum distance from x to any entity in that direction of the path. Hence, once both messages are received, x can determine r(x). It is rather straightforward to extend this method to an arbitrary tree G. In any case, the number of bits is 2(n-1) and the time is 2r(x). It is interesting to note that the use of the Transformation Lemma on the algorithm of [8] would have still resulted in an O(n) bit complexity but would have required a O(n^2) time complexity.

Example. Simultaneous Clock Reset

The local clocks in a fully synchronous system tick simultaneously but might not sign the same value. The Simultaneous Clock Reset Problem is the problem of reaching a system configuration where all local clocks sign the same value. (It is here assumed that local clocks can be reset.) Let G be a simple path as in the previous example, and let x be the (sole) entity starting the Reset process. As before, let x send a message in both directions. If each messages is let to "bounce" at both endpoints (as shown in the figure below), every entity in the path can calculate its eccentricity by counting the number of clock ticks elapsed from the time it sent a message in a direction to the time it received the same message from that direction.



It is not difficult to show that the availability of this information (the entity's eccentricity) and, more important, the timing at which it becomes available are sufficient to correctly solve the Simultaneous Clock Reset problem, which is just a different formulation of the classical Firing Squad problem. Again, the process can be easily generalized to the case of an arbitrary tree G. The total complexity is O(n) bits and O(d(G)) time.

3.3 Guessing

Guessing is another general strategy for computing a function on the input values or on the network size without transmitting any value. The basic idea is to reduce the computation to a searching process in the solution space: the desired result is guessed and the guess verified; if the verification process succeeds, the computation is completed, otherwise a new guess is made and the entire process repeated. Obviously, the main difficulty is the establishing of the guess verification mechanism; typically, this can be achieved by using (a combination of) the other two techniques. Again, we will illustrate the guessing technique by means of examples.

Example. Minimum-Finding and Election in Ring of Known Size.

Consider again the Minimum Finding problem: G is a unidirectional ring of size n (known to all entities), and each entity has a value (not necessarily unique); the goal is for all entities with smallest value to become *minimum* while all others must become *large*. Recall that, if all values are distinct, any solution to this problem is also a solution for the Election problem. Consider the following decision protocol, DECIDE(g), where g is a parameter (the "guess") available to all entities. Assume for the moment that all entities start the execution of the protocol simultaneously.

```
DECIDE(g)
If $\mathbf{i}_x \le g$ send a message and become decided, otherwise become undecided.
If a message is received while undecided, forward the message and become decided.
```

It is easy to verify that, after n time steps, one of the following two situations will occur: i) if all values are greater than g, all entities have become *undecided*; ii) if there is at least one value $\mathbf{i}_x \le g$, all entities are *decided*; furthermore, the number of bits transmitted is either zero or n, respectively. Thus, it can be verified whether a guess g on the minimum value is an under-estimate or not (situation i and ii, respectively). Thus, the smallest value can be determined by performing a sequence of guesses. g_1, g_2,... g_j; based on the outcome of the execution of DECIDE(g_j), all nodes will choose a value g_{j+1} and simultaneously perform DECIDE(g_{j+1}); the process is repeated until the minimum value is unambiguously determined. Each execution of DECIDE (i.e., each guess) requires n time units; on the other hand, the number of bits transmitted is either zero or n, depending on whether the answer is "yes" or "no", respectively; hence, the complexity of a solution algorithm can be easily derived once a solution to the guessing process is developed. This approach yields a solution to the Minimum Finding problem and (if all values are distinct) to the Election problem also if the assumption on simultaneous initiation is removed and n is known within a constant factor. In particular, this approach yields a solution with complexity $O(k\ n)$ bits and $O(n\ \mathbf{i}^{1/k})$ time for any fixed $k \geq 1$ [10]. The same technique can be applied to an arbitrary strongly connected G, and approximate knowledge of d(G) can be substituted for knowledge of n.

Example. Election in Ring of Unknown Size.

In the previous section, we have described how the "waiting protocol" provides a solution to the election problem in a ring of known size. Since n is not known, one approach is to guess its value. This can be achieved by using a monotonically increasing sequence $n_1, n_2, \ldots, n_j$ of guesses until it is determined that $n_{j-1} < n \leq n_j$, and by exploiting the fact that the Waiting Protocol with $f(\mathbf{i}_x, n_j) = 2\ \mathbf{i}_x\ n_j$ will correctly solve the election problem for such n_j. The question is how to verify the relationship between n and a guess n_j. A solution based on both waiting and counting is described in [12].

4. CONCLUDING REMARKS

Fully synchronous systems constitute an end of the spectrum of message-passing systems, fully asynchronous ones being the other end. Their inherent property is that "time" is an explicit computational tool and can be directly employed in the protocols. The open question is how to use this tool efficiently. In order to answer this question it is necessary to understand the nature of computing with time; that is, to discover the inherent limitations, characteristics and properties of synchronous systems. In this regard, among the points made in the paper, I would like to stress the following two: i) If the transmission delays are proportional to the length of the message, message sizes must be bounded (otherwise the time complexity will be unbounded); this implies that messages cannot carry input values if such values are drawn from an unbounded universe (as in the case of "id's"). ii) The communication of any amount of information can be achieved by transmitting a constant number of bounded messages (Lemmas 1 and 2); hence, any asynchronous algorithm can be transformed into a synchronous one (Transformation Lemma). This gives a benchmark against which to compare the efficiency of synchronous solutions to problems for which asynchronous protocols already exist.

In this paper, three simple techniques which use time explicitly have been introduced and briefly described: *counting*, *waiting*, and *guessing*. These techniques and the way they exploit time show how drastically different synchronous protocols can be from the asynchronous ones solving the same problem. This difference has immediate (negative) consequences for the use of synchronizers; that is, on the strategy of simulating global clock ticks in an asynchronous environment. An important motivation for synchronizers follows from the claim that it is "easier" to develop a synchronous solution than an asynchronous one; hence, the designer can develop synchronous solutions and the synchronizer will simulate the synchronous environment on which to execute the such a solution. Without entering into the substance of the claim, I would like to point out that such an approach can lead to protocols of unbounded message complexity, regardless of the efficiency of the synchronizer. In fact, every clock tick must be simulated by an execution of the synchronizer; this means that, to simulate for example the Waiting Protocol, such an approach will require a *number* of messages proportional to the entities *values* (and not of the network size).

5. REFERENCES

[1] D. Angluin, "Local and global properties in networks of processes", *Proc. 12th ACM Symp. on Theory of Computing*, April 1980, 82-93.

[2] B. Awerbuch, "Complexity of network synchronization", *J. ACM 32*, 1985, 804-823.

[3] J. Burns, "A formal model for message passing systems", TR-91, Indiana University, Sept. 1981.

[4] C. Dwork, N.A. Lynch, L. Stockmeyer, "Consensus in the presence of partial synchrony", *J. ACM 35*, 1988, 288-323.

[5] M. Fisher, N.A. Lynch, Paterson, "Impossibility of distributed consensus with one faulty precess", *J. ACM 32*, 1985, 374-382.

[6] G.N. Frederickson, N. Lynch, "The impact of synchronous communication on the problem of electing a leader in a ring", *Proc. 16th ACM Symp. Theory of Computing*, April 1984, 493-503.

[7] E. Gafni, "Improvements in the time complexity of two message-optimal election algorithms", *Proc. 4th ACM Symp. on Principles of Distributed Computing*, Aug. 1985, 175-185.

[8] E. Korach, D. Rotem, N. Santoro, "Distributed algorithms for finding centers and medians in networks", *ACM TOPLAS 6*, 1984, 380-401.

[9] L. Lamport, "Time, clocks and the ordering of events in a distributed system", *C. ACM 21*, 1978, 558-565.

[10] J. van Leeuwen, N. Santoro, J. Urrutia, S. Zaks, "Guessing games and distributed computations in anonymous networks", *Proc. 14th Int. Coll. on Automata, Languages and Programming*, July 1987, 347-356.

[11] F. Mattern, "Virtual time and global states of distributed systems", *Proc. Int. Workshop on Parallel and Distributed Algorithms*, Oct. 1988,

[12] M. Overmars, N. Santoro, "Time vs Bits: an improved algorithm for leader elections in synchronous rings", *Proc. 6h Symp. on Theoretical Aspects of Computer Science*, Feb. 1989, 282-293.

[13] J. Pachl , E. Korach, D. Rotem, "Lower bounds for distributed maximum finding algorithms, *J. ACM 31*, 1984, 380-401.

[14] N. Santoro, D. Rotem, "On the complexity of distributed elections in synchronous graphs", *Proc. 11th Int. Workshop on Graphtheoretic Concepts in Computer Science*, June 1985, 337-346.

[15] P. Spirakis, B. Tampakas, "Efficient distributed algorithms by using the Archimedean time assumption", *Proc. 5th Symp. on Theoretical Aspects of Computer Science*, Feb. 1988, 248-263

[16] P. Vitanyi, "Distributed elections in an Archimedean ring of processors", *Proc. 16th ACM Symp. Theory of Computing*, April 1984, 542-547.

Observations on ℓ-exclusion

Gary L. Peterson
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332

0. Abstract

Simple algorithms are given for the version of the mutual exclusion problem where up to ℓ of the processes may be in the critical section simultaneously. The model is concurrent processes communicating via atomic read-write shared variables. The algorithms, from simplest to most complex, are for the deadlock-free, lockout-free and FIFO-enabling forms of the problem. The latter is the main solution and is given first with unbounded size shared variables and then with bounded size variables. All algorithms remain correct even if as many as $\ell - 1$ processes fail by stopping. Large parts of these algorithms are derived by applying and generalizing previously known techniques.

1. Introduction

The standard mutual exclusion, or critical section, problem assumes that there is an activity that a group of processes must cooperate in performing so that at most one process is performing the activity at a time. Additional requirements placed on the problem typically ensure that processes have some guarantee of eventually performing the activity [Dij, Knu]. The ℓ-mutual exclusion problem (or ℓ-exclusion problem for short) assumes that upto ℓ processes may perform the activity simultaneously. One advantage of having multiple concurrent accesses is that failure by stopping of as many as $\ell - 1$ processes can be tolerated.

Fischer, Lynch, Burns and Borodin [FLBB] initially studied this problem in the generalized test-and-set model. Dolev, Gafni and Shavit [DGS] give a very complex solution for the case of read-write shared memory where the variables can be one-writer safe bits. Their algorithm, however, is only lockout-free and does not have any additional fairness properties. The main solution given here uses one-writer variables and has the FIFO-enabling property. FIFO-enabling is the generalization of FIFO to the ℓ-exclusion case. The variables can be easily simulated by one-writer safe bits using the usual constructions [Pet2, B&P1]. Afek et al. [ADGMS] report a solution to this problem but their paper was not available at the time of this writing. If the critical sections use distinct copies of different resources we then have the *slotted* ℓ-exclusion problem [B&P2]. Any of our algorithms can be combined with the slotted algorithm in [B&P2] to give a slotted solution to the ℓ-exclusion problem.

2. Definitions

The following definitions are based upon those given in [FLBB] with the change from a test-and-set model taken into account. We assume that the reader is familiar with common terms such as execution sequence (or schedule), entry protocol, exit protocol, critical section, remainder region, etc. We assume there are n processes labelled P_1 to P_n. Our first definition gives the property that must be maintained by all correct algorithms.

Definition. A protocol preserves *ℓ-exclusion* if for all points in all schedules at most ℓ processes are in their critical section states.

Our algorithms will tolerate some failure by stopping.

Definition. A process *fails* in a given schedule if it is given a finite number of turns and its last state in not in its remainder region. A schedule is t-halting if at most t processes fail.

```
SharedVariables
    Trying[1..n] range 0..2, initially 0

/* Code for Process i */
Top: Trying[i] := 1
wait until $|\{j \mid \text{Trying}[j]=2 \text{ or } j<i \text{ and Trying}[j]=1\}| < \ell$
Trying[i] := 2
repeat
    if $|\{j \mid \text{Trying}[j]=2 \text{ or } j<i \text{ and Trying}[j]=1\}| > \ell$ then goto Top
until $|\{j \mid \text{Trying}[j]=2\}| \leq \ell$
Critical Section
Trying[i] := 0
```

Figure 1. Deadlock-free ℓ-exclusion algorithm.

The next definition ensures that some process enters and leaves its critical section when one or more non-failed processes are trying.

Definition. A protocol is *ℓ-deadlock-free* if for all infinite $(\ell-1)$-halting schedules some process is executing its critical section an infinite number of time.

Unfortunately, ℓ-deadlock-free does not guarantee a particular process can enter its critical section.

Definition. A protocol is *ℓ-lockout-free* if for all infinite $(\ell-1)$-halting schedules and all points in that schedule, if a process is in its entry protocol or exit protocol, there is a later point in which it is in its critical section or remainder region, respectively.

Lockout-free is usually the strongest fairness property required, but stronger conditions can be met. The strongest condition for 1-exclusion is FIFO which is loosely defined to mean that a waiting process cannot be passed by a later trying process. With ℓ-exclusion a more complex situation arises. A process can be waiting since there are ℓ processes in their critical sections and once they leave it can be passed by a later trying process since there is room for both of them. We want the previously waiting process to be able to directly enter its critical section without any further waiting. Hence our next sequence of definitions.

Definition. A subset of states of a process is *wait-free* if, for *all* schedules where that process does not fail, once the process enters the subset of states it eventually leaves the subset of states.

Definition. A process in its entry protocol is *enabled* to enter its critical section if it can enter its critical section without waiting.

Definition. An algorithm is *FIFO-enabling* if: 1. Each process has a wait-free subset of states at the beginning its entry protocol called a *doorway*. 2. When P_i completes its doorway before P_j starts its doorway, P_i is enabled before P_j enters its critical section.

3. A simple deadlock-free ℓ-exclusion algorithm.

The simplest deadlock-free algorithm for standard mutual exclusion is due to Burns and Lynch [B&L] and requires only a single one-writer shared bit per process. Unfortunately, there seems to be no obvious generalization of their algorithm to ℓ-exclusion in general. Another simple deadlock-free algorithm is contained within Eisenberg and McGuire's lockout-free algorithm [E&M], which has been previously adapted by others such as Katseff [Kat]. It requires a 3-valued one-writer shared variable per process. By modifying the tests appropriately we have a straight-forward deadlock-free ℓ-exclusion algorithm, Figure 1.

The three values of the Trying values represent three levels of priority for trying to enter the critical section. The value 0 means a process is in its remainder region and is not interested in entering its critical section. The value 1 means a process is trying to enter its critical section but may have to wait at the first "wait" loop for processes that have a higher priority. Another process has higher priority if its Trying variable set to 2 or if its Trying variable is set to 1 and it has a lower process number. Processes cycling within the "repeat" loop have Trying set to 2. A process stays in the loop until either it can enter the critical section (when it sees that at most ℓ processes have Trying set to 2) or more than ℓ process have a higher priority. In the latter case, a process starts over. It is an interesting open question whether this problem can be solved with only two values per shared variable, even for just $n = 3$ and $\ell = 2$.

The arguments for correctness are very straightforward. For mutual exclusion, assume that somehow more than ℓ processes can enter the critical section at some point in a given execution sequence. Consider that last of those processes to set its Trying variable to 2. That process cannot enter the critical section as it will see more than ℓ processes (including itself) with their Trying variables set to 2 during the test at the end of the "repeat" loop.

For deadlock-free, assume that there is an infinite execution sequence with all non-failed trying processes unsuccessfully unable to enter their critical sections and upto $\ell - 1$ other processes having failed. It is first shown that no process can cycle forever from the top of the algorithm down through the repeat statment and back to the top via the "if" and "goto" statements if no process is passing through its critical section (and therefore setting its Trying variable to 0 repeatedly). Consider the highest numbered such cycling process. Note the complementary nature of the tests in the "wait" and "if" statements. If the test failed during the "wait" statment then it will also fail in the "if" statement for the highest numbered cycling process. Hence all non-failed trying processes can be divided into two groups: those waiting at the first "wait" loop with Trying set to 1 and those stuck in the "repeat" loop with Trying set to 2. There cannot be more than ℓ processes in the latter group (including failed processes) else at least one of the non-failed process will pass the test in the "if" statement and go back to the top (making it a cycler). If there are no non-failed processes in the latter group then there are at most $\ell - 1$ (all failed) in the group and some process in the first group will be able to exit the "wait" loop. Hence we have at least one non-failed process of a group of at most ℓ with Trying set to 2. At least one of these will therefore be able to exit the "repeat" loop and enter its critical section.

4. A simple lockout-free ℓ-exclusion algorithm.

In [Pet1] a simple two process mutual exclusion algorithm was given that has a nice generalization to n processes. That generalization can be trivially modified to solve the lockout-free case of our ℓ-exclusion problem. Recall the key idea behind the generalization is that processes proceed through levels: at most $n-1$ can succeed through the first level, at most $n-2$ through the second level, etc. In the original algorithm we go through $n-1$ levels until only one process remains. Here, we go through $n - \ell$ levels until ℓ remain. The original was not designed to handle failed processes, hence the wait condition is changed slightly so that a process only waits for other processes on the same or higher level. The lockout-free ℓ-exclusion algorithm is given in Figure 2. Note that is uses several multi-writer multi-valued shared variables as well as some 1-writer multi-valued shared variables.

The ℓ-exclusion argument trivially follows from the original algorithm's arguments hence we proceed to the lockout-free case. A process cannot wait forever at a given level while one or more processes cycle through the algorithm to the critical section and out

```
SharedVariables
    Trying[1..n] range 0..n, initially 0
    Last[1..n] range 1..n, initially 1

/* Code for Process i */
for level = 1 to n−ℓ do {
  Trying[i] := level
  Last[level] := i
  wait until |{j| Trying[j]≥level}| ≤ n−level or Last[level] ≠ i }
Critical Section
Trying[i] := 0
```

Figure 2. Lockout-free ℓ-exclusion algorithm.

else the waiting process will instead find that the Last variable for its level no longer contains its process number and proceeds to the next level. This leaves us to consider deadlock. Assume that P_i is a non-failed trying process that is the non-failed process at the highest level. It is stuck waiting if it sees more than $n-$level (which is at most ℓ) other processes at the same or higher level. At most $\ell - 1$ of these are failed so it must be waiting for one or more other non-failed processes at the same or higher level. But none can be at a higher level by the definition of P_i. Hence there is at least one other non-failed process at the same level as P_i, and furthermore they all cannot proceed to the next level. But at most one of these will see that Last[level] is its own process number, hence at least one will go onto the next level, giving us a contradiction.

Replacing the multi-writer Last variables with 1-writer variables is not an easy task, even though the write operations consist only of P_i writing i into them. It is trivial to show that such restricted multi-writer variables plus 1-writer variables are sufficient to simulate arbitrarily used multi-writer multi-valued variables. Hence, not only can the Last variables be simulated using a solution to the multi-writer concurrent reading while writing problem, e.g., [P&B], but their simulation effectively requires it. However, as the next section shows, we can solve the lockout-free (in fact FIFO-enabling) case without resorting to a multi-writer concurrent reading while writing solution.

5. *A simple unbounded FIFO-enabling ℓ-exclusion algorithm.*

The next algorithm is the preliminary form of our main algorithm. It solves the ℓ-exclusion problem with the FIFO-enabling property and uses 1-writer shared variables. Unfortunately, some of these variables are of unbounded size. The main algorithm will be along the same lines but with bounded size 1-writer shared variables. (It is becoming quite common to develop a concurrent algorithm by first prototyping it with unbounded size variables.) This algorithm is a variation of Lamport's bakery algorithm [Lam] adapted to the ℓ-exclusion problem. Each process, upon beginning its protocol, chooses a *Version Number* (given in the program as VN). It selects a Version Number that is one higher than any other process's Version Number. It then waits until it is one of the ℓ-highest trying processes in terms of priority. The priority system is the same used in [Lam]: P_j has a higher priority than P_i if VN[j]<VN[i] or if VN[j]=VN[i] and j <i. (Note: the highest priority process will have one of the smallest Version Numbers!) For 1-exclusion, what has been described so far would be an adequate solution. However, for ℓ-exclusion, an extra set of variables called Switch is needed. Each process flips its Switch variable value upon starting its protocol. Furthermore, the setting of the Switch variables has to be done at the same time a process sets Trying to true. Hence the pair (Trying,Switch) denotes a pair of Boolean values encoded into a single four-valued variable, where each

```
SharedVariables
    (Trying,Switch)[1..n] type (Boolean,Boolean), initially (false,false)
    VN[1..n] range 0.., initially 0

/* Code for Process i */
(Trying,Switch)[i] := (True, not Switch[i])
VN[i] := 1 + max{VN[j]| $1 \le j \le n$}
for j := 1 to n do SSwitch[j] := Switch[j]
/* End of doorway */
wait until |{j| Trying[j] and Switch[j]=SSwitch[j] and (VN[j],j)≤(VN[i],i)}| $\le \ell$
Critical Section
Trying[i] := false
```

Figure 3. Unbounded FIFO-enabling ℓ-exclusion algorithm.

can be read or written individually or simultaneously. The full algorithm is given in Figure 3.

The sole purpose of the Switch variables is to preserve FIFO ordering. Without them the following situation arises: P_1 goes through its doorway and is ready to perform its wait loop. P_2 goes through its doorway and performs the wait loop once. Seeing only one other process it enters its critical section. Many other processes with low Version Numbers now set their Trying variables on and then freeze. P_1 is now waiting to enter the critical section since it sees more than ℓ trying processes with lower Version Numbers. P_1 is not enabled even though the later process P_2 has entered its critical section. Hence we need some extra variables. At the end of its doorway, a process records all the Switch variables in its local SSwitch variables. As part of its wait loop it will ignore any process that has since changed its Switch variable, i.e., a later starting process. This will be seen to preserve FIFO-enabling.

Showing that ℓ-exclusion is preserved is straightforward. Assume that more than ℓ processes can somehow be in the critical section at the same time. Let P_i be the lowest priority process of that group. How could P_i have gotten in? Clearly there must be at least one other process, call it P_j, that has a higher priority but which was not included by P_i as a process to wait for in its last execution of the wait loop's test. In particular, P_j must have changed one of its shared variables between the time P_i last read it and when P_j entered its critical section. The changing variable cannot be VN[j] as Version Numbers only increase; either P_j still has a higher priority than P_i or P_i isn't the one with the lowest priority. This leaves the (Trying,Switch)[j] pair. But the last time either of these changes before the critical section is just before P_j selects a new Version Number. Hence, P_j will read VN[i] and choose a *higher* Version Number than P_i's, giving it a lower priority.

Deadlock-free is also straightforward. If there are ℓ or more processes trying to enter the critical section and none can, let P_i be the highest priority non-failed trying process. Since there are at most $\ell - 1$ failed processes, it will be one of the ℓ highest priority processes and will quickly enter the critical section.

This leaves FIFO-enabling. Suppose P_i has completed its doorway ahead of P_j and P_j has entered the critical section but P_i must wait before entering. Clearly P_i will have a higher Version Number than P_j. Which process is P_i waiting for that P_j didn't wait for? It must be some high priority process, P_k, with its Trying variable now on that had it off when P_j looked at it. But note that when setting Trying to true, Switch is also changed. So P_i will not wait for P_k unless P_k has changed its Switch variable 2 (or larger

```
type
    VersionNumber = 1..4;
    Buffer = record /* Each process will write: */
        /* Its Version Numbers, */
        VN : array [1..n] of VersionNumber;
        /* its previous Version Number, */
        PVN : array [1..n] of VersionNumber;
        /* other process's last read Version Number, */
        OVN : array [1..n] of VersionNumber;
    end;
    BufferArray : array [1..n] of Buffer;

SharedVariables
    (Trying,Switch)[1..n] : (Boolean,Boolean), initially (false,false)
    /* Main, common shared data structure for processes. */
    /* Only process i writes into Buff[i]. */
    Buff : BufferArray;
    /* Initial Version Number List. */
    /* Process i reads IOVN[1..n, i] and writes IOVN[i,1..n]. */
    IOVN : array [1..n, 1..n] of VersionNumber;

InitialValues
    Buff[1..n].VN[1..n] := 2;
    Buff[1..n].PVN[1..n] := 1;
    Buff[1..n].OVN[1..n] := 1;
    IOVN[1..n, 1..n] := 1;
Variables
    j : integer 1..n;
    SaveBuff, NewSaveBuff : BufferArray;
    NewBuff : Buffer;
    SawChanged : array [1..n] of Boolean;
    SaveIOVN : array [1..n] of VersionNumber;
```

Figure 4a. Declarations.

even) number of times since P_i looked at it at the end of P_i's doorway. But Switch is changed only once per use of the protocol, changing twice requires two separate entries to the protocol. And during the first entry P_k will sets it Version Number to be higher than P_i's. Hence P_k has a lower priority than P_i and P_i will not wait for it.

6. *The bounded FIFO-enabling ℓ-exclusion algorithm*

The multiwriter concurrent reading while writing (atomic registers) algorithm of Peterson and Burns [P&B] supplies the method which allows us to replace unbounded Version Numbers with bounded ones. The Version Numbers of [P&B] are used to order writers so that readers can select the "last" writer to have written. But the scheme can also be used to select the first ℓ with highest priority among waiting processes. In general processes read other processes' Version Numbers, select and write new Version Numbers and then wait until they are one of the ℓ lowest ranked processes. The Version Number scheme is too complex to go into here, see [P&B] for fuller details. In short, the order between processes is pairwise; if VN[j,i]=OVN[i,j] then P_i has a higher priority than P_j, i.e., P_i "saw" P_j. Another case is if P_i saw P_k but P_j didn't, etc. Processes must choose Version Numbers carefully, requiring another set of variables: IOVN. The apparent order between processes in the shared variables only reflects the recent history and is not necessarily the true order. However, keeping track of ones current (VN) and previous (PVN)

```
/* A simple non-atomic scan of the other processes. */
Procedure SimpleScan(var LocalBuff : Buffer)
Variables
    j : integer 1..n;
begin
  for j := 1 to n do
    LocalBuff[j] := Buff[j]
end SimpleScan;

/* Compare the processes' Version Numbers from two scans. */
Procedure CompareVNs(var FirstBuff, SecondBuff: Buffer;
      var NoChange, TimedOut : Boolean; var ChangedTwice: integer 1..n)
Variables
    j : integer 1..n;
begin
  for j := 1 to n do
    begin
      if FirstBuff[j].VN[1..n] $\neq$ SecondBuff[j].VN[1..n] then
        begin
          if SawChanged[j]
            then /* Saw process changed before.*/
              begin
                TimedOut := true ;
                ChangedTwice := j
              end
            else SawChanged[j] := true;
          NoChange := false;
        end
    end
end CompareVNs;
```

Figure 4b. Basic procedures.

Version Numbers suffices to maintain adequate order. For a given schedule and a point in that schedule let *rank*(i) denote P_i's actual rank and *rank*$'(i)$ be its perceived rank. There is an important relation between *rank*(i) and *rank*$'(i)$.

Remark. If *rank*$(i) \leq \ell$ then *rank*$'(i) \leq \ell$ otherwise *rank*(i)=*rank*$'(i)$.

In other words, they are the same unless a process is one of the ℓ with highest priority, in which case the relative order no longer matters. Hence Version Numbers preserve the needed priority information and can replace directly the unbounded Version Numbers of the previous section. The full algorithm is given in Figures 4a–4d. Most of Figures 4a–4c come from [P&B] and the details of those parts are explained there. Figure 4d contains the main body of the algorithm.

One of our requirements is that we can take an *atomic scan* of all processes. That is, read all the Version Number information as atomically as possible, a simpler operation than an atomic snapshot [AADGMS, And]. We, of course, cannot directly perform such an operation but we can (nearly) simulate it. The full description of our atomic scan algorithm is in [P&B]. A process can get an atomic scan by rereading over and over until it gets two consecutive identical scans or if some other process changes Version Numbers twice. The former is in effect an atomic scan. The latter causes a condition called *time-out.* If another process changes Version Numbers twice during an attempted atomic scan, that other process has passed and entered its critical section. We can let the scanning process then also enter its critical section. In effect, a passed process will enter the critical section and preserve FIFO-enabling.

```
Procedure AtomicScan(var SaveBuff : BuffArray; var TimedOut: Boolean);
Variables
     ChangedTwice : integer 1..n;
     NoChange : Boolean;
     NewSaveBuff : BuffArray;
begin
   SawChanged[1..n] := false;
   TimedOut := false;
   /* First scan. */
   SimpleScan(SaveBuff);
   repeat
      NoChange := true;
      /* Put other processes' Version Numbers on your Initial OVN list. */
      for j := 1 to n do
         IOVN[i,j] := SaveBuff[j].VN[i];
      /* Second scan and compare. */
      SimpleScan(NewSaveBuff);
      CompareVNs(SaveBuff,NewSaveBuff,NoChange,TimedOut,ChangedTwice);
      if NoChange then
         begin
            /* Third scan and compare. */
            SaveBuff := NewSaveBuff;
            SimpleScan(NewSaveBuff);
            CompareVNs(SaveBuff,NewSaveBuff,NoChange,TimedOut,ChangedTwice);
         end
      /* If tests fail then last scan becomes first scan. */
      SaveBuff := NewSaveBuff;
   until NoChange or TimedOut;
   /* Continue until no change or timed out (a process changed twice). */
end AtomicScan;
```

Figure 4c. Full process's scan algorithm.

The proofs of correctness for the bounded algorithm are mostly the same as for the unbounded case. There are two changes of note: the perceived rank not being quite the same as actual rank and that a process can time-out and enter the critical section directly. Neither cause harm as $rank(i)$ and $rank'(i)$ only differ when a process is one of the ℓ highest priority, in which case relative rank no longer matters. Entering the critical section due to time-out means that there is another process, that has a *lower* priority than the timed-out process, went into the critical section. A full proof must wait for a much longer version of the paper.

The bounded FIFO-enabling algorithm also has some additional properties not required by the basic problem. The exit protocol is wait-free (and in fact just a single statement). Also, a process past its doorway never waits for any later starting process, a stronger condition than FIFO-enabling in certain ways.

By generalizing the Version Number scheme slightly, it is possible to order any bounded size set of recent events in a distributed system, i.e., simulate time stamps [DS].

7. References.

[AADGMS] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merrit, and N. Shavit, "Atomic snapshots of shared memory," *Proceedings of the 9th Ann. ACM Symp. on Principles of Dist. Computing*, 1990, pp. 1-14.

```
/* Algorithm for process i: */
(Trying,Switch)[i] := (True, not Switch);
/* Read other processes' Initial OVN list. */
for j := 1 to n do
   SaveIOVN[j] := IOVN[j,i];
AtomicScan(SaveBuff; TimedOut);
if TimedOut then go to CS
for j := 1 to n do
   begin
      /* Get the other processes' Version Numbers. */
      NewBuff.OVN[j] := SaveBuff[j].VN[i];
      /* Set Previous Version Numbers. */
      NewBuff.PVN[j] := SaveBuff[i].VN[j];
      /* Select a new Version Number. */
      /* (It can't be current VN, VN on others OVN list,
         or VN on others Initial OVN list.)*/
      NewBuff.VN[k]:= any({1,2,3,4}–
            {SaveBuff[i].VN[j], SaveBuff[j].OVN[i], SaveIOVN[j]});
   end;
/* Write new version numbers. */
Buff[i] := NewBuff
for j := 1 to n do SSwitch[j] := Switch[j]
/* End of doorway */
repeat
   AtomicScan(SaveBuff; TimedOut);
   if TimedOut then go to CS
   with SaveBuff do
   rank := |{j| OVN[i,j] = VN[j,i] or
         OVN[j,i] $\notin$ {OVN[i,j],PVN[i,j]} or
         $\exists$ k OVN[j,k]=VN[k,j] and OVN[i,k]$\neq$VN[k,i] or
         $\forall$ k OVN[i,k]=VN[k,i] $\Rightarrow$ OVN[j,k]=VN[k,j] and j<i }
until rank $\leq \ell$
CS: Critical Section
Trying[i] := false
```

Figure 4d. Top level algorithm.

[ASGMS] Y. Afek, D. Dolev, E. Gafni, M. Merrit, and N. Shavit, "A bounded first-in first-enabled solution to the ℓ-exclusion problem," 4th Inter. Workshop on Dist. Algorithms, 1990

[And] J. Anderson, "Composite registers," *Proceedings of the 9th Ann. ACM Symp. on Principles of Dist. Computing*, 1990, pp. 15-30.

[B&L] J. Burns and N. Lynch, "Mutual exclusion using indivisible reads and writes," *Proc. 18th Annual Allerton Conf. on Comm., Control, and Computing*, 1980.

[B&P1] J. Burns and G. Peterson, "Constructing multi-reader atomic values from non-atomic values," *Proceedings of the 6th Ann. ACM Symp. on Principles of Dist. Computing*, 1987 pp. 222-231

[B&P2] J. Burns and G. Peterson, "The ambiguity of choosing," *Proceedings of the 8th Ann. ACM Symp. on Principles of Dist. Computing*, 1989, pp. 145-157.

[DGS] D. Dolev, E. Gafni, and N. Shavit, "Toward a non-atomic era: ℓ-exclusion as a test case," *Proceedings of the 20th Ann. ACM Symp. on Theory of Computing*, 1988, pp. 78-92.

[D&S] D. Dolev and N. Shavit, "Bounded concurrent time-stamp systems are constructible," *Proceedings of the 20th Ann. ACM Symp. on Theory of Computing*, 1989, pp 454-466

[Dij] E.W. Dijkstra, "Solution of a problem in concurrent programming control," *Comm. ACM 8*, 1965, pg. 569.

[E&M] M. Eisenberg and M. McGuire, "Further comments on Dijkstra's concurrent programming control problem," *Comm. ACM 15*, 1972, pg. 999.

[FLBB] M. Fischer, N. Lynch, J. Burns and A. Borodin, "Resource allocation with immunity to limited process failure," *Proceedings of the 20th Ann. IEEE Symp. on Foundations of Comp. Sci*, 1979, pp. 234-254.

[Kat] H.P. Katseff, "A new solution to the critical section problem," *Proceedings of the 9th Ann. ACM Symp. on Theory of Computing*, 1978, pp 86-88.

[Knu] D. Knuth "Additional comments on a problem in concurrent programming control," *Comm. ACM 9*, 1966, pg. 321.

[Lam] L. Lamport, "A new solution of Dijkstra's concurrent control problem," *Comm. ACM 17*, 1974, pg. 453-455.

[Pet1] G. Peterson, "Myths about the mutual exclusion problem," *IPL 12*, **3**, 1981, pp. 115-116

[Pet2] G. Peterson, "Concurrent reading while writing," *ACM Trans. on Prog. Lang. and Systems 5*, **1**, 1983, 46-55

[P&B] G. Peterson and J. Burns, "Concurrent reading while writing II: the multi-writer case," *Proceedings of the 28th Ann. IEEE Symp. on Foundations of Comp. Sci.*, 1987, pp. 383-392.

Efficiency of Semi-Synchronous versus Asynchronous Networks

Hagit Attiya*
Computer Science Department
Technion, Haifa 32000, Israel

Marios Mavronicolas
Aiken Computation Laboratory
Harvard University, Cambridge, MA 02138

SEPTEMBER 1990

Abstract

The *s-session problem* is studied in the *asynchronous* and the *semi-synchronous* networks. Processes are located at the nodes of an undirected graph G and communicate by sending messages along links that correspond to the edges of G. It is assumed that the (real) time for message delivery is at most d. In the asynchronous model the time between any two consecutive steps of any process is in the interval $[0, 1]$; in the semi-synchronous model the time between any two consecutive steps of any process is in the interval $[c, 1]$, where $0 < c \leq 1$, the *synchronous* model being a special case where $c = 1$.

For the asynchronous model, an upper bound of $diam(G)(d+1)(s-1)$ and a lower bound of $diam(G)d(s-1)$ are presented, where $diam(G)$ is the *diameter* of G. For the semi-synchronous model, an upper bound of $1 + \min\{\lfloor \frac{1}{c} \rfloor + 1, diam(G)(d+1)\}(s-2)$ is presented. The main result of the paper is a lower bound of $1 + \min\{\lfloor \frac{1}{2c} \rfloor, diam(G)d\}(s-2)$ for the time complexity of any semi-synchronous algorithm for the s-session problem under the assumption that $d \geq \frac{1-c}{c}$. These results imply a time separation between semi-synchronous (in particular, synchronous) and asynchronous networks.

---

*This work was performed when the author was with Laboratory for Computer Science, MIT, supported by ONR contract N00014-85-K-0168, by NSF grant CCR-8915206 and by DARPA contract N00014-87-K-0825.

1 Introduction

Most distributed systems are based on a *communication network*—a collection of n processes arranged at the nodes of an undirected graph G and communicating by sending messages across links of this graph. Central to the programming of distributed systems are *synchronization* problems, where a process is required to guarantee that all processes have performed a particular set of steps. Naturally, the timing information available to processes has critical influence on the time complexity of synchronization.

Arjomandi, Fischer and Lynch ([1]) introduced the *session problem* to study the impact of timing information on the time complexity of synchronization. Roughly speaking, a *session* is a sequence of events that contains *at least* one step by each process. An algorithm for the s-*session problem* guarantees that each execution of the algorithm includes at least s separate sessions. It was assumed in [1] that processes communicate via *shared variables* and the time complexity of the session problem was studied in *synchronous* and *asynchronous* models. Informally, in a *synchronous* system processes operate in lock-step, taking steps simultaneously, while in *asynchronous* systems processes work at completely independent rates and have no way to estimate time. The results of [1] show that the problem can be solved much faster in the synchronous model than in the asynchronous one.

In reality, however, there is an important middle ground: in most distributed systems, processes neither work at lock-step nor at a completely independent rate. For example, processes may have access to inaccurate clocks that operate at approximately, but not exactly, the same rate. We model these *semi-synchronous* systems by assuming that there exist a lower and an upper bound on processes' step time that enable processes to estimate time. We address the cost of synchronization in *communication networks* under the semi-synchronous and the asynchronous models by studying the session problem. We present upper and lower bounds for the time complexity of solving the s-session problem in the asynchronous and semi-synchronous model, thus separating the semi-synchronous (in particular, synchronous) from the asynchronous networks.

Time is measured under the following assumptions on the system: Messages sent on a communication link incur a delay in the range $[0, d]$, for some parameter $d \geq 0$. In the *asynchronous* model processes have rate $[0, 1]$, while in the *semi-synchronous* model processes have rate $[c, 1]$, for some parameter c such that $0 < c \leq 1$.[1] We assume that all processes take a step at time 0.

Our upper bound results are as follows. Our first algorithm relies on explicit communication to ensure that the needed steps have occured and does not use any timing information. In the asynchronous model, this algorithm has time complexity $diam(G)(d+1)(s-1)$, where the *diameter*, $diam(G)$, of an undirected graph G is the *maximum* distance between any two nodes. In the semi-synchronous model, this algorithm can be modified to take advantage of the initial synchronization and achieve a slightly better time complexity of $1 + diam(G)(d+1)(s-2)$. The second algorithm does not use any communication and relies only on timing information; this algorithm works only in the semi-synchronous model. The time complexity of this algorithm is $1 + (\lfloor \frac{1}{c} \rfloor + 1)(s-2)$. These algorithms can be combined to yield a semi-synchronous algorithm for the s-session problem whose time complexity is $1 + \min\{\lfloor \frac{1}{c} \rfloor + 1, diam(G)(d+1)\}(s-2)$.

---

[1] Note that the synchronous model is a special case of the semi-synchronous model where $c = 1$.

Our lower bound results are as follows. For the asynchronous model, we prove an almost matching lower bound of $diam(G)d(s-1)$ for the time complexity of any algorithm for the s-session problem. For the semi-synchronous model, we prove two lower bounds. We first show a simple lower bound of $\lfloor\frac{1}{c}\rfloor(s-2)$ for the case where communication is not used. We then present our main result: a lower bound of $1+\min\{\lfloor\frac{1}{2c}\rfloor, diam(G)d\}(s-2)$ for the time complexity of any semi-synchronous algorithm for the s-session problem; this result relies on the assumption that $d \geq \frac{1-c}{c}$.

The lower bounds presented in this paper use the same general approach as in [1]. However, the lower bound proof in [1] uses fan-in arguments, while our lower bounds are based on information propagation arguments using long delays of messages, combined with appropriate selection of processes and careful timing arguments. The asynchronous lower bound is based and improves on a result of Lynch ([6]) showing a lower bound of $rad(G)d(s-1)$.[2]

Awerbuch ([5]) has introduced the concept of a *synchronizer* as a way to translate algorithms designed for synchronous networks to asynchronous networks. Although the results of [5] may suggest that any synchronous network algorithm can be translated into an asynchronous algorithm with constant time overhead, our results imply that this is *not* the case: for some values of the parameters, any translation of a semi-synchronous (in particular, synchronous) algorithm for the s-session problem *must* incur a non-constant time overhead.

Due to lack of space, the details of some of our definitions, constructions and proofs are omitted here; we refer the reader to [4] for an expanded version of this paper.

2 Definitions

In this section, we present the definitions for the underlying formal model,[3] define what it means for an algorithm to solve the *s-session problem* and introduce some notation.

A *system* consists of n processes $p_1, \ldots, p_n$. Processes are located at the nodes of a graph $G = (V, E)$, where $V = [n]$. For simplicity, we identify processes with the nodes they are located at and we refer to nodes and processes interchangeably. Each process p_i is modelled as a (possibly infinite) state machine with state set Q_i. The state set Q_i contains a distinguished *initial state* $q_{0,i}$. The state set Q_i also includes a subset I_i of *idle* states. We assume that any state of p_i includes a special component, $buffer_i$, which is p_i's message buffer. A *configuration* is a vector $C = (q_1, \ldots, q_n)$ where q_i is the local state of p_i; denote $state_i(C) = q_i$. The *initial configuration* is the vector $(q_{0,1}, \ldots, q_{0,n})$. Processes communicate by sending *messages* (taken from some possibly infinite alphabet $\mathcal{M}$) to each other. A *send action* $send(j, m)$ represents the sending of message m to a neighboring process p_j. Let $\mathcal{S}_i$ denote the set of all send actions $send(j, m)$ for all $m \in \mathcal{M}$ and all $j \in [n]$, such that $(i, j) \in E$.

We model computations of the system as sequences of *atomic events*, or simply *events*, for short. Each event is either a *computation event*, representing a computation step of a single process, or a *delivery event*, representing the delivery of a message to a process. Each *computation event* is specified by $comp(i)$ for some $i \in [n]$. In the computation step associated with event $comp(i)$, the process p_i, based on its local state, changes its local state and performs some set S of send actions, where S is a finite subset of $\mathcal{S}_i$. Each delivery event has the form $del(i, m)$ for some $m \in \mathcal{M}$. In a delivery step associated with the event $del(i, m)$, the message m is added to $buffer_i$, p_i's message buffer.[4] We say that a computation step is *non-idle* if it is taken by a process in a non-idle state.

---

[2] The *radius*, $rad(G)$ of G is the *minimum*, over all nodes in V of the maximum distance from that node to any other node in V. For any undirected graph G, $rad(G) \leq diam(G) \leq 2rad(G)$.

[3] These definitions could be expressed in terms of the general *timed automaton model* described in [8, 3, 7].

Each process p_i follows a deterministic local algorithm $\mathcal{A}_i$ that determines p_i's local computation, i.e., the messages to be sent and the state transition to be performed. More specifically, for each $q \in Q_i$, $\mathcal{A}_i(q) = (q', S)$ where q' is a state and S is a set of send actions. We assume that once a process enters an idle state, it will remain in an idle state, i.e., if q is an idle state, then q' is an idle state. An *algorithm* (or a *protocol*) is a sequence $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ of local algorithms.

An *execution* is an infinite sequence $\alpha = C_0, \pi_1, C_1, \ldots, \pi_j, C_j, \ldots$, of alternating configurations and events that satisfies the following conditions: the changes in processes' states are according to the transition function; only a process which takes a step or to which a message is delivered changes its state; each sending of a message is matched to a later message delivery and each message delivery to an earlier send. We adopt the convention that finite prefixes of an execution end with a configuration, and denote the last configuration in a finite execution prefix α by $last(\alpha)$.

A *timed event* is a pair (t, π), where t, the "time", is a nonnegative real number, and π is an event. A *timed sequence* is an infinite sequence of alternating configurations and timed events $\alpha = C_0, (t_1, \pi_1), C_1, \ldots, (t_j, \pi_j), C_j, \ldots$, where the times are nondecreasing and unbounded. We define timed executions as follows. Fix real numbers c and d, where $0 \leq c \leq 1$ and $0 \leq d < \infty$. Letting α be a timed sequence as above, we say that α is a *timed execution* of $\mathcal{A}$ provided that the following all hold: $C_0, \pi_1, C_1, \ldots, \pi_j, C_j, \ldots$ is an execution of P; there are computation events for all processes at time 0; the time between any two consecutive events of the same process is in the interval $[c, 1]$; the time between a message sending event and the corresponding message delivery event is at most d.

We say that α is an *execution fragment* of $\mathcal{A}$ if there is an execution α' of $\mathcal{A}$ of the form: $\alpha' = \beta\alpha\beta'$. This definition is extended to apply to timed executions in the obvious way. For a finite execution fragment $\alpha = C_0, (t_1, \pi_1), C_1, \ldots, (t_k, \pi_k), C_k$, we define $t_{start}(\alpha) = t_1$ and $t_{end}(\alpha) = t_k$.

The *asynchronous* model is defined by taking $c = 0$, while the *semi-synchronous* model is defined by taking $0 < c \leq 1$; the *synchronous model* is a special case of the latter where $c = 1$ and $d < 1$. To simplify the expression of our time bounds in terms of the parameters, d, and c, we sometimes approximate the bounds in the case that $1 \ll d$. For example, in this case we have $\Delta \approx d$.

An execution fragment $C_1, \pi_1, C_2 \ldots, \pi_m, C_m$ is a *session* if for each i, $i \in [n]$, there exists at least one event $\pi_j = comp(i)$, for some $j \in [m]$, which is a non-idle step of the underlying execution. Intuitively, a session is an execution fragment in which each process takes at least one non-idle step. An execution α *contains s sessions* if it can be partitioned into at least s separate execution fragments such that each of them is a session. These definitions are extended to apply to timed executions in the obvious way.

---

[4] The system model can be extended to allow arbitrary state change upon message delivery without changing the results; for clarity of presentation, we chose not to do so.

An algorithm *solves the s-session problem within time t* (on G) if each of its timed executions α satisfies the following: α includes s sessions and all processes are in an idle state in α after time t.

Consider an undirected graph $G = (V, E)$. For any $i, j \in V$, let $dist(i,j)$ be the *distance* of i and j in G, i.e., the number of edges in the shortest path in G from i to j. The *diameter* of G, $diam(G)$, is the maximum distance between any two nodes in V. A node $i \in V$ is a *peripheral* node of G if $\max_{j \in V} dist(i,j) = diam(G)$; informally, a peripheral node "realizes" the diameter of G. A node $j \in V$ is *antipodal* to a node $i \in V$ if $dist(i,j) = \max_{k \in V} dist(i,k)$.

3 Upper Bounds

3.1 The Asynchronous Model

We start with a simple asynchronous algorithm in which processes communicate in order to learn about completion of a session before proceeding to the next session. We prove:

Theorem 3.1 *Let G be any graph. There exists an asynchronous algorithm $\mathcal{A}^{as}$ that solves the s-session problem on G within time $diam(G)\Delta(s-1)$.*

Proof: We describe an asynchronous algorithm, $\mathcal{A}^{as}$, that solves the s-session problem within time $diam(G)\Delta(s-1)$. Each process maintains as part of its state a variable that gives its current session number; upon hearing that every other process has reached its current session, it increases its session number by 1 and notifies all other processes about that. Notification is done by sending messages to them along a shortest-path tree rooted at it. (The details of how this is done can be found in, e.g., [9].) The process enters an idle state when its session number is set to s. In the full paper we prove that in any execution of $\mathcal{A}^{as}$ there are at least s sessions and all processes enter an idle state no later than $diam(G)\Delta(s-1)$. Thus, $\mathcal{A}^{as}$ solves the s-session problem within time $diam(G)\Delta(s-1)$. ■

3.2 The Semi-Synchronous Model

In the semi-synchronous model we can slightly improve $\mathcal{A}^{as}$ by taking advantage of the available initial synchronization; specifically, each process performs exactly as in $\mathcal{A}^{as}$, except that it does not wait to hear that every process has completed its first session, but passes directly to the second one upon taking its second step. We have:

Theorem 3.2 *Let G be any graph. There exists a semi-synchronous algorithm $\mathcal{A}_1^{ss}$ that solves the s-session problem on G within time $1 + diam(G)\Delta(s-2)$.* ■

Roughly speaking, an algorithm $\mathcal{A}$ *uses no communication* if no process state transition can result in a send action. We describe next a semi-synchronous algorithm that uses no communication and, thus, prove that the timing information available in the semi-synchronous model can be exploited to obtain a bound which is sometimes better than the previous bound. More specifically, we show:

Theorem 3.3 *Let G be any graph. There exists a semi-synchronous algorithm $\mathcal{A}_2^{ss}$ which solves the s-session problem on G within time* $1 + (\lfloor \frac{1}{c} \rfloor + 1)(s-2)$.

Proof: We describe a semi-synchronous algorithm, $\mathcal{A}_2^{ss}$, which solves the s-session problem on G within time $1 + (\lfloor \frac{1}{c} \rfloor + 1)(s-2)$. Each process maintains as part of its state a variable with the number of computation steps it has taken so far. It enters an idle state when this variable is set to $1 + (\lfloor \frac{1}{c} \rfloor + 1)(s-2)$. In the full paper we prove that in any execution of $\mathcal{A}_2^{ss}$ there are at least s sessions and all processes enter an idle state no later than $1 + (\lfloor \frac{1}{c} \rfloor + 1)(s-2)$. Thus, $\mathcal{A}_2^{ss}$ solves the s-session problem within time $1 + (\lfloor \frac{1}{c} \rfloor + 1)(s-2)$. ■

By running the algorithms of Theorems 3.1 and 3.3 "side by side," halting when the first of them does, we get:

Theorem 3.4 *Let G be any graph. There exists a semi-synchronous algorithm $\mathcal{A}^{ss}$ which solves the s-session problem on G within time* $1 + \min\{\lfloor \frac{1}{c} \rfloor + 1, diam(G)\Delta\}(s-2)$.

4 Lower Bounds

In all of our lower bounds, we use an infinite timed execution in which processes take steps in round-robin order, starting with p_1, at rate 1 and all messages are delivered after exactly d delay; we will call it a *slow, synchronous* timed execution.

4.1 The Asynchronous Model

We start by showing that for the asynchronous model, the algorithm presented in Theorem 3.1 is optimal. The proof of the following theorem is based on delaying information propagation and then perturbing an execution to obtain an execution of the algorithm which does not include s sessions.

Theorem 4.1 *Let G be any graph. There does not exist an asynchronous algorithm which solves the s-session problem on G within time strictly less than* $diam(G)d(s-1)$.

Proof: Assume, by way of contradiction, that there exists an algorithm, $\mathcal{A}$, which solves the s-session problem within time strictly less than $diam(G)d(s-1)$. We construct a timed execution of $\mathcal{A}$ which does not include s sessions. Consider a slow, synchronous infinite timed execution $\gamma = \alpha\alpha'$ of $\mathcal{A}$, where α is the longest prefix of γ such that some process is not in an idle state in $last(\alpha)$. We reorder events in α to obtain another timed execution prefix β, such that $\beta\alpha'$ is a timed execution of $\mathcal{A}$ that does not include s sessions.

By assumption, $t_{end}(\alpha) < diam(G)d(s-1)$. Write $\alpha = \alpha_0\alpha_1 \ldots \alpha_{s-1}$, where $\alpha_0 = \emptyset$ and for each k, $1 \leq k \leq s-1$, $t_{end}(\alpha_k) - t_{end}(\alpha_{k-1}) < diam(G)\Delta$. (We adopt the convention that: $t_{end}(\alpha_0) = 0$.) For some sequence $i_0, \ldots, i_{s-1}$ of peripheral nodes, we construct from each execution fragment α_k an execution fragment $\beta_k = \rho_k\sigma_k$, such that ρ_k contains no computation step of $p_{i_{k-1}}$ and σ_k contains no computation step of p_{i_k}.

For each k, $1 \leq k \leq s-1$, we show how to construct β_k inductively. For the base case, let i_0 be an arbitrary peripheral node of G, and take β_0 to be the empty execution. Assume we have picked $i_0, \ldots, i_{k-1}$ and constructed $\beta_0, \ldots, \beta_{k-1}$; we now show how to pick i_k and create β_k. Let i_k be some node that is antipodal to i_{k-1}, i.e., $dist(i_{k-1}, i_k) = diam(G)$; note that i_k is also peripheral. For any node u, ρ_k includes all events at u that occur until (and including) time $t_{end}(\alpha_{k-1}) + dist(u, i_{k-1})d$ in α_k; all these events are scheduled to occur in the same order as in α_k and all occur at time 0 in ρ_k. σ_k will include all events of u that occur strictly after time $t_{end}(\alpha_{k-1}) + dist(u, i_{k-1})d$ in α_k; their order will be preserved and all occur at time 0. It is easy to see that for each k, $1 \leq k \leq s-1$, $\beta_k = \rho_k \sigma_k$ has the desired properties.

In the full paper we show that each receiving event is after the corresponding sending event in β. Since all events in β occur at time 0 and there are no lower bounds on either process step time or message delivery time in the asynchronous model, lower and upper bounds on step time and message delivery time are preserved in β. This establishes that β is a timed execution of $\mathcal{A}$. We also show that there are at most $s-1$ sessions in β; however, in α' all processes are in an idle state, so there is no additional session in α'. A contradiction. ∎

We remark that the general outline of this lower bound proof follows [1, 6]. Our improvement is achieved by being careful and choosing only peripheral nodes in the construction of β.

4.2 The Semi-Synchronous Model

In previous sections, we have seen two algorithms that solve the s-session problem in the semi-synchronous model. The first of them, $\mathcal{A}_1^{ss}$, solves the s-session problem within time $1 + diam(G)\Delta(s-2)$. Designed for the asynchronous model, $\mathcal{A}_1^{ss}$ has the interesting property that processes do not use any timing information. Loosely speaking, the lower bound proved in Theorem 3.1 says that if processes have no timing information, then $diam(G)d(s-1)$ is a lower bound for any algorithm which solves the s-session problem on G. Recall, also, that $\mathcal{A}_2^{ss}$ uses no communication but relies only on timing information to achieve an upper bound of $1 + (\lfloor \frac{1}{c} \rfloor + 1)(s-2)$. We first show that this upper bound is close to optimal in the absence of communication:

Theorem 4.2 *Let G be any graph. There does not exist a semi-synchronous algorithm which solves the s-session problem on G within time strictly less than $\lfloor \frac{1}{c} \rfloor (s-2)$ and uses no communication.*

Proof: Assume, by way of contradiction, that there exists an algorithm $\mathcal{A}$ that solves the s-session problem within time strictly less than $\lfloor \frac{1}{c} \rfloor (s-2)$, and uses no communication. We construct a timed execution of $\mathcal{A}$ which does not include s sessions.

Let α be a slow, synchronous infinite timed execution of $\mathcal{A}$. Assume, without loss of generality, that p_1 is the last process to enter an idle state in α. Let $\alpha = \alpha_0 \alpha'$, where α_0 includes events at time 0, while α' is the remaining part of α. Let m be the number of non-idle steps taken by any process in α'. It must be that $m < \lfloor \frac{1}{c} \rfloor (s-2)$, since $\mathcal{A}$ solves the s-session problem within time $< \lfloor \frac{1}{c} \rfloor (s-2)$, and α is slow. Now modify α' to get a new execution β in which all processes except p_n, operate at the fastest rate, i.e., at rate c. This can be done since there are no receive events in α. In β, all processes but p_n enter an idle state at time $c \cdot m < c\lfloor\frac{1}{c}\rfloor(s-2) \leq c\frac{1}{c}(s-2) = s-2$. Thus, p_n performs strictly less than $s-2$ steps when all other processes are not in an idle state. Therefore, at most $s-2$ sessions can be completed in β; hence, at most $s-1$ sessions can be completed in $\alpha_0\beta$. A contradiction. ■

We show next that communication and timing information cannot be combined to get an upper bound that is significantly better than the upper bound achieved in Theorem 3.4. We prove:

Theorem 4.3 *Let G be any graph and assume that $d \geq \frac{1-c}{c}$. There does not exist a semi-synchronous algorithm which solves the s-session problem on G within time strictly less than* $1 + \min\{\lfloor\frac{1}{2c}\rfloor, diam(G)d\}(s-2)$.

Proof: Assume, by way of contradiction, that there exists an algorithm, $\mathcal{A}$, which solves the s-session problem within time strictly less than $1 + \min\{\lfloor\frac{1}{2c}\rfloor, diam(G)d\}(s-2)$. We construct a timed execution of $\mathcal{A}$ which does not include s sessions. Denote $b = \min\{\lfloor\frac{1}{2c}\rfloor, diam(G)d\}$.

Consider a slow, synchronous infinite timed execution γ of $\mathcal{A}$. Assume $\gamma = \beta_0\alpha\alpha'$, where β_0 contains only events that occur at time < 1 and $\beta_0\alpha$ is the longest prefix of γ such that some process is not in idle state in $last(\beta_0\alpha)$. Denote $T = t_{end}(\beta_0\alpha)$ and note that $T < 1 + b(s-2)$, or $\frac{T-1}{s-2} < b$. This implies that there exists some $\epsilon > 0$ such that $\frac{T-1}{s-2} < b - \epsilon$. Denote $s' = \lceil\frac{T-1}{b-\epsilon}\rceil$ and note that $s' \leq \lceil s-2\rceil = s-2$.

We write $\alpha = \alpha_1\alpha_2\ldots\alpha_{s'}$, where for each k, $1 \leq k \leq s'$, α_k contains all events that occur at t, where $1 + (k-1)(b-\epsilon) \leq t < 1 + k(b-\epsilon)$, while $\alpha_{s'}$ contains all events occurring at t, where $1 + (s'-1)(b-\epsilon) \leq t \leq T$. Essentially, we partition α in execution fragments $\alpha_1, \alpha_2, \ldots, \alpha_{s'}$ each of which is completed within time $b - \epsilon < b$. We reorder events in α to obtain a timed execution that contains at most s' sessions.

For some sequence $i_0, \ldots, i_{s'}$ of peripheral nodes, we construct from each execution fragment α_k an execution fragment $\beta_k = \rho_k\sigma_k$, such that ρ_k contains no computation step of $p_{i_{k-1}}$ and σ_k contains no computation step of p_{i_k}. For each k, $1 \leq k \leq s'$, we construct β_k inductively. For the base case, let i_0 be an arbitrary peripheral node of G. Assume we have picked $i_0, \ldots, i_{k-1}$ and constructed $\beta_1, \ldots, \beta_{k-1}$ we now show how to pick i_k and create β_k. Let i_k be some node that is antipodal to i_{k-1}, i.e., $dist(i_{k-1}, i_k) = diam(G)$; note that i_k is also peripheral. For any process p_j, ρ_k includes all computation and message delivery events at p_j that occur at time $< 1 + (k-1)(b-\epsilon) + dist(j, i_{k-1})d$ in α_k; they will occur in the same order as in α_k. The first of the computation events at p_j occurs at $t_{end}(\beta_{k-1}) + c$. On the other hand, σ_k will include all computation and message delivery events at j that occur at time $\geq 1 + (k-1)(b-\epsilon) + dist(j, i_{k-1})d$; they occur in the same order as in α_k. In both ρ_k and σ_k, computation events will occur at a rate of c and each message delivery event that occurs between two consecutive computation events in α_k, occurs after and at exactly the same time as the first of them in σ_k. Notice that for each k, $1 \leq k \leq s'$, $\beta_k = \rho_k\sigma_k$ has the desired properties. The first computation step of i_{k-1} in σ_r occurs at time $t_{start}(\sigma_k) = t_{end}(\rho_k)$. All other later steps of i_{k-1} in σ_k occur at a rate of c; the last of them occurs at $t_{end}(\sigma_k)$. For each process p_j, we schedule each computation step, π_j, of p_j in σ_k to occur simultaneously with a computation step, $\pi_{i_{k-1}}$, of i_{k-1} which is such that π_j and $\pi_{i_{k-1}}$ occurred at the same time in α_k.

In the full paper we show that in β each receiving event is after the corresponding sending event, lower and upper bound on step time are preserved and the time between a message sending event and the corresponding message delivery event is at most d. This establishes that $\beta_0\beta\alpha'$ is a timed execution of $\mathcal{A}$. We can also show that β contains at most $s' \leq s-2$ sessions; also, in β_0 there is exactly one session. Hence, there are at most $s-1$ sessions in $\beta_0\beta$. However, in α' all processes are in an idle state, so there is no additional session in α'. A contradiction. ■

5 Discussion and Future Research

Assuming that $1 \ll d$, i.e., that $\Delta \approx d$, we have presented matching upper and lower bounds of $diam(G)\Delta(s-1)$ for the asynchronous model. For the semi-synchronous model, we have proved an upper bound of $1+\min\{\lfloor\frac{1}{c}\rfloor+1, diam(G)\Delta\}(s-2)$ and a lower bound of $1+\min\{\frac{1}{2c}, diam(G)\Delta\}(s-2)$. Neglecting roundoffs, the upper bound is within a factor of 2 of the lower bound.

The work presented in this paper continues the study of time bounds in the presence of timing uncertainty within the framework of the semi-synchronous model ([2, 3]). Our results give a time separation between semi-synchronous (in particular, synchronous) and asynchronous networks.

Our work leaves open several interesting problems. An obvious open problem is to close the gap between the lower and the upper bounds for the semi-synchronous case. We have assumed processes start synchronously at time 0; how do the bounds change if this assumption is removed? The definition of a session does not require processes to be "aware" of a session's end; what happens if the definition is changed? Another interesting direction is to study problems in the semi-synchronous model, assuming that processes communicate via shared-memory (as in [1]).

Acknowledgements:

We wish to thank Nancy Lynch for making [6] available to us and for helpful discussions. We also like to thank Pino Persiano for comments on an earlier version of the paper. The second author would like to thank Harry Lewis for his constant advice and encouragement.

References

[1] E. Arjomandi, M. Fischer and N. Lynch, "Efficiency of Synchronous versus Asynchronous Distributed Systems," *Journal of the ACM*, Vol. 30, No. 3 (1983), pp. 449–456.

[2] H. Attiya, C. Dwork, N. Lynch and L. Stockmeyer, "Bounds on the Time to Reach Agreement in the Presence of Timing Uncertainty," in preparation.

[3] H. Attiya and N. Lynch, "Time Bounds for Real-Time Process Control in the Presence of Timing Uncertainty," in *Proceedings of the 10th Real-Time Systems Symposium*, pp. 268–284, December 1989. Expanded version: Technical Memo MIT/LCS/TM-403, Laboratory for Computer Science, MIT, July 1989.

[4] H. Attiya and M. Mavronicolas, "Efficiency of Semi-Synchronous versus Asynchronous Networks," TR 21-90, Aiken Computation Laboratory, Harvard University.

[5] B. Awerbuch, "Complexity of Network Synchronization," *Journal of the ACM*, Vol. 32, No. 4 (1985), pp. 804–823.

[6] N. Lynch, *Asynchronous vs. Synchronous Computation* (guest lecture), lecture notes for MIT 18.438: Network Protocols and Distributed Graph Algorithms, Spring 1989.

[7] N. Lynch and H. Attiya, "Using Mappings to Prove Timing Properties," in *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing* , pp. 265-280, August 1990. Expanded version: Technical Memo MIT/LCS/TM-412.b, Laboratory for Computer Science, MIT, December 1989.

[8] M. Merritt, F. Modugno and M. Tuttle, "Time Constrained Automata," manuscript, November 1988.

[9] A. Segall, "Distributed Network Protocols," *IEEE Transactions on Information Theory*, Vol. IT-29, No. 1 (January 1983), pp. 23–35.

Clock Synchronization and the Power of Broadcasting

Joseph Y. Halpern
IBM Almaden Research Center
San Jose, CA 95120

Ichiro Suzuki
Department of Electrical Engineering and Computer Science
University of Wisconsin – Milwaukee
Milwaukee, WI 53201

Abstract: We investigate the power of a broadcast mechanism in a distributed network. We do so by considering the problem of synchronizing clocks in an error-free network, under the assumption that there is no upper bound on message transmission time, but that broadcast messages are guaranteed to be received within an interval of size ϵ, for some fixed constant ϵ. This is intended to be an idealization of what happens in multiple access networks, such as the Ethernet. We then consider tradeoffs between the type and number of broadcasts, and the tightness of synchronization. Our results include (1) matching upper and lower bounds of $(1+\frac{1}{K})\epsilon$ on the precision of clock synchronization attainable for $n \geq 3$ processes using K $(n-1)$-casts for $3 \leq K \leq n$, (2) matching upper and lower bounds of $(1+\frac{1}{n})\epsilon$ on the precision of clock synchronization attainable for $n \geq 3$ processes using an arbitrary number of $(n-1)$-casts, and (3) matching upper and lower bounds of $(1+\frac{n-2}{n})\epsilon$ on the precision attainable using 2-casting.

1 Introduction

Broadcasting is a basic primitive that is available in many computer networks (such as the Ethernet [MB76]). In some cases, the network provides the user with the ability to send an ℓ-cast, that is, a broadcast message to some subset of size ℓ. In this paper, we investigate the power of broadcasting—both by comparing broadcasting to point-to-point message transmission and by comparing ℓ-casting to k-casting, for $\ell \neq k$—in the context of the problem of synchronizing clocks in a fully-connected, error-free network. (We could, of course, identify a point-to-point message with a broadcast to one process, but we find it useful to distinguish broadcasts from point-to-point messages.)

Even in an Ethernet-style network, where all messages can be viewed as broadcast messages, an ℓ-cast can be viewed as being cheaper than an k-cast for $\ell < k$, since if a process is not among the recipients of a broadcast, it must do significantly less processing of the message. In particular, in large networks, we can view a broadcast to a large subset of processes as being considerably more expensive than a point-to-point message. As Gray says [Gr88], "Practitioners want architectures which scale to arbitrary size networks. As a consequence they limit attention to *multicast*: broadcast to a "small" group of processes [Ch88]. Multicast scales to large networks ... [whereas broadcast does not, because of its cost]." (We remark that one of the uses of multicasting in the V system is to transmit clock synchronization information.) Given these observations, it becomes important to understand the tradeoffs between the use of broadcast messages and the ability to achieve certain goals in distributed systems.

We have taken clock synchronization as a paradigm problem to study in this context because it has been so well studied, both in the presence of faults (see, for example, [DHS86, HSSD84, LM85, LL88, ST87]) and without faults [HMM85, La78, LL84, SS88]. The basic problem we consider is the same as that considered in [HMM85, LL84]: we assume that each process has a clock that runs at the rate of real time, and the problem is to synchronize these clocks as tightly as possible. Of course, in practice, clocks drift apart. However, the drift is typically small, and by ignoring it here, we can examine the impact of broadcasting more carefully. In all the papers cited above with the exception of [SS88], it is assumed that processes can communicate only by sending point-to-point messages. Moreover, it is assumed that there is a (commonly) known upper bound on the message transmission time. Here we drop the assumption that there is an upper bound on the message transmission time. Without further assumptions, it is easy to show that clock synchronization is impossible; i.e., there is no upper bound on how tightly we can synchronize clocks. (We provide a formal proof of this fact in the next section.) So, following [SS88], we assume that the network has a broadcast primitive, in addition to a facility providing point-to-point message transmission. Although there is no upper bound on how long it will take for a broadcast message to arrive, we do assume that all processes receive a broadcast message within a (real time) window of ϵ, for some (presumably small) constant ϵ. This is meant to be an idealized version of what actually happens in an Ethernet (when probabilistic considerations are ignored). Processors can try to broadcast, but if there is a conflict (when two or more processes try to broadcast at the same time), there must be a retry. There is no upper bound on the number of retry attempts. Although with high probability a broadcast will succeed after very few attempts, we cannot place an upper bound on message transmission time. On the other hand, if a broadcast does succeed, then all the intended recipients receive it within a small window of time.

In [SS88], the problem of clock synchronization in this model was considered under the extra assumption that a process can broadcast to everyone but itself. An upper bound of $(1+\frac{1}{n})\epsilon$ on the precision of synchronization was proved, as well as an almost matching lower bound of $(1+\frac{1}{n(n+2)})\epsilon$. Here we extend and generalize the results of [SS88], investigating in more detail the tradeoffs between the type and number of broadcasts allowed and the precision of synchronization attainable. Among other results, we show that if precisely K $(n-1)$-casts are allowed, then we can attain a synchronization of $(1+\frac{1}{K})\epsilon$ for $3 \leq K \leq n$; if $K=2$, then we can only attain a synchronization of 2ϵ. We prove matching lower bounds in all these cases. Moreover, we show that even with an arbitrary number of $(n-1)$-casts, we can only synchronize to within $(1+\frac{1}{n})\epsilon$; that is, we can get optimal synchronization with n $(n-1)$-casts. This result is explained by our observation that what really matters is not how many $(n-1)$-casts are used, but the number of distinct subsets of processes that are recipients of $(n-1)$-casts. Our lower bound of $(1+\frac{1}{K})\epsilon$ holds no matter how many $(n-1)$-casts are used (and no matter how much point-to-point communication is used) as long as there are no more than K distinct subsets of processes that receive broadcast messages. Since there are n distinct subsets of size $(n-1)$, our lower bound of $(1+\frac{1}{n})\epsilon$ holds with arbitrary $(n-1)$-casts. This proves that the upper bound of [SS88] is in fact optimal.

We then turn our attention to ℓ-casting. In the case of n-casting, where a broadcast reaches all processes in the system including the sender, we can prove a tight bound of ϵ on the precision of synchronization attainable. For $2 \leq \ell \leq n-1$, we provide an algorithm using ℓ-casting that synchronizes to within $(1+\frac{n-\ell}{n})\epsilon$ for $2 \leq \ell \leq n-1$. We conjecture that this is optimal, although we have only been able to prove this for the case $\ell = 2$ and $\ell = n-1$. For $2 < \ell < n-1$, we do have a nontrivial lower bound of $\max(\frac{n-1}{n}\frac{\ell}{\ell-1}\epsilon, (1+\frac{1}{\ell+1})\epsilon)$.

The rest of this paper is organized as follows. In the next section we give the necessary definitions and show that broadcasts are necessary to achieve synchronization in networks where there is no upper bound on message transmission time. We consider $(n-1)$-casts in Section 3, proving tight bounds on the synchronization achievable with K $(n-1)$-casts. In Section 4 we consider ℓ-casting. We conclude with some open questions and discussion in Section 5. In this abstract, we present only the key proofs and ideas; all the details and the missing proofs can be found in the full paper.

2 Preliminaries

As we mentioned above, the framework we use is essentially that of [HMM85, LL84]. Consider a network with $n \geq 2$ processes, say $P_1, P_2, \ldots, P_n$. Process P_i has a *physical clock* C_i which is a real-valued function of real time. We assume that the physical clocks run at the rate of real time and they cannot be reset by the processes; that is, $C_i(t) = C_i(0) + t$ at all times t. The processes have no access to the real time. Process P_i has a local variable A_i (for *adjustment*) which provides the difference between the logical and physical clocks of P_i. That is, the *logical clock* $L_i(t)$ of process P_i at time t is given by $L_i(t) = C_i(t) + A_i(t)$, where $A_i(t)$ is the value of A_i at t.

We assume that at some fixed real time (say, in response to some external signal) all the processes wish to synchronize their clocks as tightly as possible. They do so by sending messages, both point-to-point messages from one process to another process and broadcasts from some process to a subset consisting of at least 2 processes. There is no upper bound on the time it takes a message to arrive from the time that it is sent; however, broadcasts arrive to all their intended recipients within a window of size ϵ; i.e., if a broadcast is sent at time t, there is some $t' \geq t$ such that all recipients receive the broadcast within the interval $[t', t' + \epsilon]$. We shall be interested in the size of the subset of recipients; an *ℓ-cast* is a broadcast from a process to a set of ℓ processes.

Following [HMM85], we assume that a clock synchronization algorithm is a deterministic algorithm in which the state transition and the action of sending messages of process P_i at time t is determined only by the value of $C_i(t)$ and the history of P_i at t. Here, the *history* of P_i at t is the sequence consisting of tuples of the forms $\langle P_j, m, T, y\rangle$ for every message P_i has sent or received before t, where $\langle P_j, m, T, y\rangle$ represents that message m was either sent ($y =$ **sent**) or received ($y =$ **received**) to or from P_j when the value of C_i was T.[1]

An algorithm is said to *synchronize* the logical clocks to within γ if the algorithm eventually terminates, and when it terminates at time t, $|L_i(t) - L_j(t)| \leq \gamma$ holds for all $i \neq j$. Although we do not require that the algorithm enforce monotonicity (i.e., that $L_i(t') \geq L_i(t)$ if $t' \geq t$), since this makes our lower bounds stronger, in fact, all our algorithms do enforce monotonicity.

An execution of an algorithm can be viewed as a function s such that for all $1 \leq i \leq n$ and times t, $s(i,t)$ is the history of P_i at t during the execution. This observation leads us to the following definitions. A function s such that $s(i,t)$ is a pair consisting of a history of P_i and a clock reading T for all $1 \leq i \leq n$ and times t is called a *scenario.* A scenario s is *legal* with respect to a given algorithm if there exists an execution of the algorithm such that for all $1 \leq i \leq n$ and times t, the history in $s(i,t)$ is the history of P_i at time t and the clock reading T in $s(i,t)$ is $C_i(t)$.

Let s_1 be a scenario of a given algorithm. We say that scenario s_2 is obtained from s_1 by *shifting* P_i by ρ_i if $s_2(i,t) = s_1(i, t-\rho_i)$ for $1 \leq i \leq n$. Intuitively, state transitions and sending and receiving of messages of P_i occur uniformly ρ_i later in s_2 than in s_1. Using $C_{i,s}(t)$ (resp. $L_{i,s}(t)$) to denote the physical (resp. logical) clock reading of process P_i in scenario s, note that we have $C_{i,s_2}(t) = C_{i,s_1}(t) - \rho_i$. Since the clock adjustment made by a process in an execution of the algorithm depends only on the reading of its physical clock and its message history, it is easy to see that the following lemma holds.

Lemma 1 *Let s_1 and s_2 be scenarios which are legal with respect to a given algorithm, where s_2 is obtained from s_1 by shifting P_i by ρ_i for $1 \leq i \leq n$. Let t be any time after the execution of the algorithm terminates at every process in both s_1 and s_2. Then $L_{i,s_2}(t) = L_{i,s_1}(t) - \rho_i$.*

We conclude this section by proving, as claimed in the introduction, that we really need to use broadcast messages in order to synchronize clocks at all in our framework, given that we have no upper bound on message transmission time. In fact, we show something even stronger, namely, that every process must receive a broadcast message in order to guarantee synchronization.

---

[1] We assume that processes are up throughout the running of the algorithm. We could assume instead that processes either wake up in response to some external signal or as a result of receiving a message. We have ignored the issue of processes waking up for ease of exposition; all our results hold with essentially no change in the more complicated model.

Theorem 1 *The logical clocks of $n \geq 2$ processes cannot be synchronized by an algorithm $\mathcal{A}$ such that in each execution of $\mathcal{A}$, some process does not receive a broadcast message.*

Proof Suppose that there exists an algorithm $\mathcal{A}$ which synchronizes the logical clocks to within γ, and in each execution of $\mathcal{A}$, some process does not receive a broadcast message. We derive a contradiction, using the by-now standard "many scenarios" technique [DHS86, FLM86, HMM85, LL84].

Choose $D > \gamma$. Let s_1 be a scenario that corresponds to an execution of $\mathcal{A}$ in which the transmission time of every message (both broadcast and point-to-point) is $2D$. Suppose without loss of generality that P_1 does not receive any broadcast messages in s_1. Let s_2 be the scenario obtained from s_1 by shifting P_1 by $2D$ and all other processes by zero. Scenario s_2 is legal, since (1) all messages are received in nonnegative time, and (2) broadcast messages are received simultaneously. Specifically, the message transmission time in s_2 is zero from P_1 to $P_2, \ldots, P_n$, $4D$ from $P_2, \ldots, P_n$ to P_1, and $2D$ for all other messages.

Let t be a real time such that the algorithm terminates at every process in both s_1 and s_2 by time t. Let $L_{1,s_1}(t) = T_1$ and $L_{2,s_1}(t) = T_2$. By Lemma 1, $L_{1,s_2}(t) = T_1 - 2D$ and $L_{2,s_2}(t) = T_2$. By the assumption on γ, we have $T_1 \leq T_2 + \gamma$ and $T_2 \leq T_1 - 2D + \gamma$. By adding the two inequalities we obtain $\gamma \geq D$. This is a contradiction. □

3 Optimal clock synchronization using $(n-1)$-casting

In this section we focus on optimal clock synchronization using $(n-1)$-casting. We assume $n \geq 3$ (since if $n = 2$, then $(n-1)$-casting amounts to point-to-point communication, so that clock synchronization is impossible by Theorem 1). Informally, this is meant to correspond to the case where a process can broadcast to all processes other than itself (although, in fact, our arguments hold even in the case where a process can include itself among the recipients of its broadcast, as long as the list of recipients has no more than $n-1$ processes). Since we assume a broadcast is expensive, we focus here on the tradeoff between the number of broadcasts and the precision of synchronization.

We know from Theorem 1 that we require at least two $(n-1)$-casts in order to achieve synchronization. If we are allowed no more than two $(n-1)$-casts, then we can synchronize processes to within 2ϵ:

Theorem 2 *The logical clocks of $n \geq 3$ processes can be synchronized to within 2ϵ using two $(n-1)$-casts and one point-to-point message.*

Proof Since the algorithm is simple, we only describe it informally. We let P_1 and P_2 broadcast to $P_2, P_3, \ldots, P_n$ and $P_1, P_3, P_4, \ldots, P_n$, respectively. (The broadcast of P_2 can actually be a 2-cast to P_1 and P_3.) Processes $P_2, P_3, \ldots, P_n$ set their logical clocks to zero upon receipt of the broadcast of P_1 (that is, P_i sets $A_i = -C_i$). Then the logical clocks of $P_2, \ldots, P_n$ are synchronized to within ϵ. P_3 then sends a message to P_1 carrying the difference D between the times it received P_1's broadcast and P_2's broadcast. If P_1 received P_2's broadcast and P_3's message at physical times T and T', respectively, then P_1 sets its logical clock to $T' - T + D$ when it receives P_3's message (that is, P_1 sets $A_1 = D - T$). It is easy to show that the logical clocks of P_1 and P_2 are now synchronized to within ϵ, and hence the logical clocks of all processes are synchronized to within 2ϵ. □

It turns out that the one point-to-point message used in our algorithm is necessary. We return to this issue at the end of the section.

The upper bound of Theorem 2 is tight. The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than 2ϵ by using two $(n-1)$-casts, no matter how much additional point-to-point communication we use. The lower bound is an easy corollary to the following result, which shows that the lower bound holds even if we allow an arbitrary number of $(n-1)$-casts, as long as they cannot be sent to more than two distinct subsets of processes of size $n-1$.

Theorem 3 *The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than 2ϵ by an algorithm $\mathcal{A}$ which uses $(n-1)$-casting and point-to-point communication, if, in each execution of $\mathcal{A}$, $(n-1)$-casts cannot be sent to more than two distinct subsets of processes of size $n-1$.*

Proof Again, we use a many-scenarios argument; we omit details here. □

Corollary 1 *The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than 2ϵ by using only two $(n-1)$-casts and additional point-to-point communication.*

We now turn our attention to the case where we allow K $(n-1)$-casts, for $3 \leq K \leq n$. Again, we can prove tight bounds in this case. We start with the upper bound. Our algorithm for the upper bound is a generalization of that given in [SS88] for the case $K = n$.

Theorem 4 *The logical clocks of $n \geq 3$ processes can be synchronized to within $(1 + \frac{1}{K})\epsilon$ by using K $(n-1)$-casts for $3 \leq K \leq n$.*

Proof In the full paper we provide an algorithm that synchronizes clocks to within the required precision. The algorithm uses an averaging process much in the same spirit of the algorithm in [LL84]; we omit details here. □

We now prove a matching lower bound. Again, it turns out that the key issue is not how many $(n-1)$-casts are actually sent, but the number of distinct subsets of processes of size $n-1$ that receive $(n-1)$-casts.

Theorem 5 *The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than $(1 + \frac{1}{K})\epsilon$ by using $(n-1)$-casts and additional point-to-point communication, if $(n-1)$-casts cannot be sent to more than K distinct subsets of processes of size $n-1$, for $3 \leq K \leq n$.*

Proof The proof again uses the many-scenario technique, although we have to work a bit harder to construct the scenarios in this case. Fix an algorithm which synchronizes the logical clocks to within γ without sending $(n-1)$-casts to more than K distinct subsets of processes of size $n-1$. Without loss of generality let s_0 be a scenario in which

1. each of $P_{K+1}, \ldots, P_n$ receives all $(n-1)$-casts,
2. the transmission time of every point-to-point message is $(1 + \frac{1}{K})\epsilon$, and
3. the transmission time of broadcast messages is as given below.

Think of processes $P_1, \ldots, P_K$ as being on a ring, and define the "distance" from P_i to P_j $(1 \leq i, j \leq K)$ on the ring by:

$$d(i,j) = \begin{cases} j - i & \text{if } i \leq j \\ K + j - i & \text{if } i > j \end{cases}$$

For each $1 \leq i \leq K$, let S_i be the set of all processes other than P_i. Then in s_0, for each $1 \leq i \leq K$, the transmission time $\tau_{S_i,j}$ of the message to P_j $(j \neq i)$ in every $(n-1)$-cast to the processes in S_i from any process is:

$$\tau_{S_i,j} = \begin{cases} (2 - \frac{d(i,j)}{K})\epsilon & \text{if } 1 \leq j \leq K \text{ and } j \neq i \\ 2\epsilon & \text{if } K+1 \leq j \leq n \end{cases}$$

Note that $\tau_{S_i,j} \geq (1 + \frac{1}{K})\epsilon$ for any $i \neq j$. Clearly s_0 is legal.

Let s_1 be the scenario obtained from s_0 by shifting each P_j by $\rho_{1,j}$, where:

$$\rho_{1,j} = \begin{cases} 0 & \text{if } j = 1 \\ (1 + \frac{1}{K})\epsilon & \text{if } j = 2 \\ \frac{j-1}{K}\epsilon & \text{if } 3 \leq j \leq K \\ 0 & \text{if } K+1 \leq j \leq n \end{cases}$$

Now we show that s_1 is legal. First, note that $|\rho_{1,j} - \rho_{1,k}| \leq (1 + \frac{1}{K})\epsilon$ for any $1 \leq j, k \leq n$. This, together with the fact that the transmission time of every message in s_0 is at least $(1 + \frac{1}{K})\epsilon$, ensures that all messages have nonnegative transmission time in s_1. Next, suppose that for some $1 \leq i \leq K$, a process $(n-1)$-casts to the processes in S_i at time t in s_0. By the definition of s_0 and the construction of s_1, the messages are received by P_j $(j \neq i)$ at $t + \tau_{S_i,j} + \rho_{1,j}$ in s_1, where by elementary analysis we have the following. For $i = 1$:

$$\tau_{S_1,j} + \rho_{1,j} = \begin{cases} 3\epsilon & \text{if } j = 2 \\ 2\epsilon & \text{if } 3 \leq j \leq n \end{cases}$$

For $i = 2$:

$$\tau_{S_2,j} + \rho_{1,j} = \begin{cases} (1 + \frac{1}{K})\epsilon & \text{if } j = 1 \\ (2 + \frac{1}{K})\epsilon & \text{if } 3 \leq j \leq K \\ 2\epsilon & \text{if } K+1 \leq j \leq n \end{cases}$$

For $3 \leq i \leq n$:

$$\tau_{S_i,j} + \rho_{1,j} = \begin{cases} (1 + \frac{i-1}{K})\epsilon & \text{if } j = 1 \\ (2 + \frac{i-1}{K})\epsilon & \text{if } j = 2 \\ (1 + \frac{i-1}{K})\epsilon & \text{if } 3 \leq j \leq i-1 \\ (2 + \frac{i-1}{K})\epsilon & \text{if } i+1 \leq j \leq K \\ 2\epsilon & \text{if } K+1 \leq j \leq n \end{cases}$$

Thus the messages are received within an interval of ϵ. Therefore s_1 is legal.

We construct scenarios $s_2, \ldots, s_K$ in a completely symmetrical manner; for $2 \leq m \leq K$, s_m is the scenario obtained from s_0 by shifting each P_j by $\rho_{m,j}$, where:

$$\rho_{m,j} = \begin{cases} \frac{d(m,j)}{K}\epsilon & \text{if } 1 \leq j \leq m \\ (1 + \frac{1}{K})\epsilon & \text{if } j = m+1 \\ \frac{d(m,j)}{K}\epsilon & \text{if } m+2 \leq j \leq K \\ 0 & \text{if } K+1 \leq j \leq n \end{cases}$$

We omit the proof of the legality of $s_2, \ldots, s_K$, since the argument is similar to that for s_1.

Let t be any time after the algorithm terminates at every process in $s_0, s_1, \ldots, s_K$. Let $L_{i,s_0}(t) = T_i$ for $1 \leq i \leq K$. By Lemma 1, for $1 \leq m \leq K-1$ we have $L_{m,s_m}(t) = T_m$ and $L_{m+1,s_m}(t) = T_{m+1} - (1 + \frac{1}{K})\epsilon$, while $L_{K,s_K}(t) = T_K$ and $L_{1,s_K} = T_1 - (1 + \frac{1}{K})\epsilon$. By the assumption on γ, we have

$$T_m \leq T_{m+1} - (1 + \frac{1}{K})\epsilon + \gamma$$

for $1 \leq m \leq K-1$ and

$$T_K \leq T_1 - (1 + \frac{1}{K})\epsilon + \gamma.$$

By adding the K inequalities we obtain

$$\gamma \geq (1 + \frac{1}{K})\epsilon.$$

□

Of course, from Theorem 5, we immediately obtain:

Corollary 2 *The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than $(1 + \frac{1}{K})\epsilon$ by using K $(n-1)$-casts and additional point-to-point communication, for $3 \leq K \leq n$.*

Corollary 3 *The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than $(1 + \frac{1}{n})\epsilon$ by using $(n-1)$-casts and additional point-to-point communication.*

Proof Since there are only n distinct subsets of size $n-1$, the result follows immediately from Theorem 5. □

We remark that Corollary 3 improves the lower bound of $(1 + \frac{1}{n(n-2)})\epsilon$ proved in [SS88].

The theorems we have proved so far completely characterize the precision of synchronization achievable as a function of the number of $(n-1)$-casts. In our analysis, we have essentially ignored the point-to-point communication required, since we have viewed point-to-point communication as being essentially free compared to the cost of an $(n-1)$-cast. In the case of $K = 2$, our algorithm uses one point-to-point message in addition to the two $(n-1)$-casts. As we now show, this point-to-point transmission is essential.

Theorem 6 *The logical clocks of $n \geq 3$ processes cannot be synchronized using two $(n-1)$-casts and no point-to-point messages.*

Proof Suppose that $\mathcal{A}$ is an algorithm that synchronizes the processes to within γ and uses no more two $(n-1)$-casts and no point-to-point communication in every execution. Choose $D > \gamma$, and consider an execution where message transmission time for all messages is $2D$. Let s_1 be the scenario corresponding to this execution. Clearly every process must be the recipient of at least one broadcast message. (If not, suppose that P_1 did not receive any broadcast messages and shift P_1 by $2D$ to obtain an easy contradiction.) Since there are only two $(n-1)$-casts altogether, this means that two of the processes, say P_1 and P_2, are the recipients of only one $(n-1)$-cast each. For definiteness, suppose that P_1 is a recipient of a broadcast by P_i and P_2 is the recipient of a broadcast by P_j. Without loss of generality assume further that in s_1, process P_j sent its broadcast at or before the time that P_i sent its broadcast, which means that P_j sent its broadcast before P_i's broadcast was received. Now let s_2 be the scenario where (a) P_1 is shifted by $2D$, (b) if $i \neq 1$, then the broadcast from P_i that is received by P_1 takes time $4D$ to be delivered to each process, and (c) if $j = 1$, then the broadcast from P_1 that is received by P_2 takes time 0 to be delivered to each process. Clearly s_2 is legal. Since P_j sent its broadcast in s_1 before receiving P_i's broadcast, it must be the case that P_j sends the same broadcast at the same time in s_2 and s_1; i.e., the delay in receiving P_i's broadcast in s_2 does not affect P_j's sending its broadcast. As a consequence, it is not hard to see that P_1 and P_2 cannot distinguish s_1 and s_2. The other processes may be able to distinguish s_1 from s_2, but since there is no further communication, they cannot tell P_1 and P_2 about it. Thus P_1 and P_2 perform the same adjustments in s_1 and s_2. We can now conclude by a similar argument to that of Theorem 1 that $\gamma \geq D$, giving us contradiction. We leave details to the reader. □

In our algorithm that synchronized to within $(1 + \frac{1}{K})\epsilon$ using K $(n-1)$-casts, we used $2n-2$ point-to-point transmissions ($n-1$ in each of phases 2 and 3). We do not know to what extent this number can be reduced, although it is not hard to show that we can replace $(n-1)$ of the point-to-point transmissions by one $(n-1)$-cast. We leave the minimization of the number of point-to-point transmissions as an open problem.

4 Clock synchronization using ℓ-casting

In this section, we consider the tradeoffs that arise if we allow ℓ-casting, for arbitrary values of ℓ, rather than restricting attention to $(n-1)$-casts. We start by considering the case of n-casting.

Theorem 7 *The logical clocks of n processes can be synchronized to within ϵ using one n-cast. They cannot be synchronized any closer than ϵ using any number of broadcasts and point-to-point transmissions.*

Proof The upper bound is trivial. One process, say P_1 sends an n-cast. All processes set their logical clocks to 0 on receipt of the n-cast. Clearly they are synchronized to within ϵ. We prove the lower bound using another many-scenarios argument; we omit details here. □

We now turn our attention to ℓ-casting with $2 \le \ell \le n-1$. We again restrict attention to $n \ge 3$, since otherwise, with $\ell \le n-1$, we do not have any broadcast messages, so we know by Theorem 1 we cannot synchronize at all. Using $C(n,\ell)$ to denote n choose ℓ (i.e., $\frac{n!}{(n-\ell)!\,\ell!}$), we can generalize the upper bound argument of Theorem 4 to get:

Theorem 8 *The logical clocks of $n \ge 3$ processes can be synchronized to within $(1+\frac{n-\ell}{n})\epsilon$ by using $C(n,\ell)$ ℓ-casts and $2n-2$ point-to-point messages, for $2 \le \ell \le n-1$.*

We conjecture that the upper bound attained in Theorem 8 is tight, but we have only been able to prove this thus far for the case $\ell = 2$ and $\ell = n-1$. The case $\ell = n-1$ follows from Theorem 5. The case $\ell = 2$ follows from the following theorem.

Theorem 9 *The logical clocks of n processes cannot be synchronized any closer than $(1+\frac{n-2}{n})\epsilon$ using 2-casting and any number of point-to-point transmissions.*

Proof Fix an algorithm which synchronizes the logical clocks to within γ. Let s_0 be a scenario in which

1. in every 2-cast to P_i and P_j where $i < j$, the transmission time of the messages to P_i and P_j is 2ϵ and 3ϵ, respectively, and
2. the transmission time of every point-to-point message is 2ϵ.

Clearly s_0 is legal. Now for each $1 \le m \le n-1$, let s_m be the scenario obtained from s_0 by shifting each of $P_1, \ldots, P_m$ by 2ϵ and not shifting the other processes at all. It is easy to check that in s_m, $1 \le m \le n-1$, we have

1. the transmission time of every message (both broadcast and point-to-point) is nonnegative,
2. in every 2-cast to P_i and P_j where $1 \le i < j \le m$ or $m+1 \le i < j \le n$, P_i receives the message ϵ earlier than P_j, and
3. in every 2-cast to P_i and P_j where $1 \le i \le m$ and $m+1 \le j \le n$, P_i receives the message ϵ later than P_j.

Thus s_m is a legal scenario.

Let t be any time after the algorithm terminates at every process in $s_0, s_1, \ldots, s_{n-1}$. Let $L_{i,s_0}(t) = T_i$ for $1 \le i \le n$. By Lemma 1, for $1 \le m \le n-1$ we have $L_{m,s_m}(t) = T_m - 2\epsilon$ and $L_{m+1,s_m}(t) = T_{m+1}$. By the assumption on γ, we have

$$T_1 \le T_n + \gamma$$

and

$$T_{m+1} \leq T_m - 2\epsilon + \gamma$$

for $1 \leq m \leq n-1$. By adding the n inequalities we obtain

$$\gamma \geq (1 + \frac{n-2}{n})\epsilon.$$

□

Although we cannot prove a matching lower bound to the upper bound of Theorem 8 for $2 < \ell < n-1$, we can prove some nontrivial lower bounds.

Theorem 10 *For $2 < \ell < n-1$, the logical clocks of n processes cannot be synchronized any closer than $\max(\frac{n-1}{n}\frac{\ell}{\ell-1}\epsilon, (1+\frac{1}{\ell+1})\epsilon)$ using ℓ-casting and any number of point-to-point transmissions.*

Proof The lower bound of $(1+\frac{1}{\ell+1})\epsilon$ follows using essentially the same proof as the lower bound for $(n-1)$-casts in Theorem 7. The only difference is that we treat processes $P_{\ell+1}, \ldots, P_n$ as one process, and then use the lower bound for ℓ-casting with $\ell+1$ processes. If an ℓ-cast is sent to a subset that includes k processes among $P_{\ell+1}, \ldots, P_n$, messages to all these processes take exactly the same amount of time as it would take a message to reach process $P_{\ell+1}$ if there were exactly $\ell+1$ processes in the system. The scenario ends up looking as if an $(\ell-k+1)$-cast were sent instead of an ℓ-cast. We leave details to the reader. The proof of the lower bound of $\frac{n-1}{n}\frac{\ell}{\ell-1}\epsilon$ is similar to that of the lower bound for 2-casts in Theorem 9. The only difference is that we start with a scenario s_0 in which

1. in every ℓ-cast to $P_{i_1}, P_{i_2}, \ldots, P_{i_\ell}$ where $i_1 < i_2 < \ldots < i_\ell$, the transmission time of the message to P_{i_j}, $1 \leq j \leq \ell$, is $\frac{\ell+j-1}{\ell-1}\epsilon$, and
2. the transmission time of every point-to-point message is $\frac{\ell}{\ell-1}\epsilon$,

and then for each $1 \leq m \leq n-1$, we let s_m be the scenario obtained from s_0 by shifting each of $P_1, \ldots, P_m$ by $\frac{\ell}{\ell-1}\epsilon$ and not shifting the other processes at all. The legality of the scenarios $s_0, s_1, \ldots, s_{n-1}$ is immediate, and the rest of the argument is basically the same as that in the proof of Theorem 9. We leave details to the reader. □

5 Concluding remarks

We have investigated the power of broadcasting in the context of clock synchronization. We have not solved (and, in fact, have not closely investigated) all the questions in the area. For example, one question we have not considered in the context of ℓ-casting is the number of ℓ-casts and point-to-point transmissions required in order to achieve optimal synchronization. Notice that the algorithm in Theorem 8 required $C(n,\ell)$ ℓ-casts. We conjecture that these ℓ-casts are all required.

However, our primary goal has not been to solve all the problems that can be solved in the area, but rather to demonstrate the tradeoffs between the type and number of broadcasting, and the precision of synchronization attainable. Our results suggest, for example, that for values of ℓ for which $(n-1)$-casting is not much more expensive than ℓ-casting, $(n-1)$-casting is preferable for achieving clock synchronization. If broadcasting is expensive, then our results suggest that 2-casting might be the method of preference.

Recently, Dolev and Dwork have observed analogous tradeoffs in the context of Byzantine agreement [DD90]. They consider primitive they call a *conference call of size k*, where for any set S of k participants, whenever any one participant p utters a message m, the statement "p sent m" becomes common knowledge among the members of S. Thus, a conference call of size

k is essentially a k-cast where all messages are guaranteed to arrive simultaneously. Dolev and Dwork show that more faults can be tolerated by using *conference calls of size* k than by using conference calls of size ℓ for $k > \ell$. It would be interesting to understand these tradeoffs for other application areas.

Acknowledgments: We would like to thank Vassos Hadzilacos for his helpful comments on an earlier version of this paper.

References

[Ch88] D. Cheriton, "The V distributed system," *Communications of the ACM* **31**:3, 1988, pp. 314–332.

[DD90] D. Dolev and C. Dwork, "On-the-fly generation of names and communication primitives," unpublished manuscript, 1990.

[DHS86] D. Dolev, J. Halpern and R. Strong, "On the possibility and impossibility of achieving clock synchronization," *J. Computer and System Sciences* **32**, 1986, pp. 230–250.

[FLM86] M. J. Fischer and N. A. Lynch and M. Merritt, "Easy impossibility proofs for distributed consensus problems," *Distributed Computing* **1**:1, 1986, pp. 26–39.

[Gr88] J. N. Gray, "The Cost of Messages," *Proc. of the 7th ACM Symp. on Principles of Distributed Computing*, 1988, pp. 1–7.

[HMM85] J. Halpern, N. Megiddo and A. Munshi, "Optimal precision in the presence of uncertainty," *Journal of Complexity* **1**, 1985, pp. 170–196.

[HSSD84] J. Halpern, B. Simons, R. Strong and D. Dolev, "Fault-tolerant clock synchronization," *Proc. 3rd Annual ACM Symposium on Principles of Distributed Computing*, Vancouver, Canada, 1984, pp. 89–102.

[La78] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM* **21**:7, 1978, pp. 558–565.

[LM85] L. Lamport and P. M. Melliar-Smith, "Synchronizing clocks in the presence of faults," *Journal of the ACM* **32**:1, 1985, pp. 52–78.

[LL84] J. Lundelius and N. Lynch, "An upper and lower bound for clock synchronization," *Information and Control* **62**, 1984, pp. 190–204.

[LL88] J. Lundelius and N. Lynch, "A new fault-tolerant algorithm for clock synchronization," *Information and Computation* **77**, 1988, pp. 1–36.

[MB76] R. Metcalf and D. Boggs, "Ethernet: distributed packet switching for local computer networks," *Communications of the ACM* **19**:7 1976, pp. 395–404.

[ST87] T. K. Srikanth and S. Toueg, "Optimal clock synchronization," *Journal of the ACM* **34**:3, 1987, pp. 626–645.

[SS88] K. Sugihara and I. Suzuki, "Nearly optimal clock synchronization under unbounded message transmission time," *Proc. 1988 International Conference on Parallel Processing* **III**, St. Charles, Illinois, 1988, pp. 14–17.

A NEW ALGORITHM FOR THE RECOVERY OF IMAGES FROM THEIR TRIPLE-CORRELATION FUNCTIONS[1]

A. Hammoud[1], T. J. Schulz[2], and D. L. Snyder[1]
[1]Electronic Systems and Signals Research Laboratory
Washington University, St. Louis, MO
and
[2]Environmental Research Institute of Michigan
Ann Arbor, MI

ABSTRACT

A number of methods exist in the literature for recovering intensity images of objects using triple-correlation functions or their Fourier transforms, called bispectra [1,2]. We present an iterative method for recovering images given their triple-correlation function. This work extends that of T. Schulz and D. Snyder [3] to accommodate the recovery of an object's intensity when its triple-correlation function is known exactly or when this function is estimated from quantum-limited data. Our results will be compared with those obtained by the recursive [1] and least squares [2] methods. Our method is capable of recovering the image intensity up to shifts and 180° rotations when the triple-correlation function is known exactly and to a better approximation than the alternative methods when only quantum-limited data are available.

REFERENCES

1. A. W. Lohman, G. Weigelt, and B. Wirnitzer, "Speckle masking in astronomy: triple-correlation theory and applications," *Applied Optics*, Vol. 22, pp. 4028-4037, 1983.

2. J. C. Maron, P. P. Sanchez, and R. C. Sullivan, "Unwrapping algorithm for least-squares phase-recovery from the modulo 2π bispectrum phase," *J. Optical Society of America A*, Vol. 7, pp. 14-20, January 1990.

3. T. J. Schulz and D. L. Snyder, "Imaging a randomly moving object from quantum-limited data: applications to image recovery from second and third order autocorrelations," *J. Optical Society of America A*, in review, 1990.

---

[1] This work was supported by the National Science Foundation under Grant MIP-8722463.

IMAGE RESTORATION ALGORITHMS USING MODIFIED QUADRATIC DISCRIMINANT FUNCTIONS

Sung-Jea Ko, Thomas M. Forest, and Fumitaka Kimura
Department of Electrical and Computer Engineering
University of Michigan-Dearborn
Dearborn, MI 48128

In this paper, a bank of filters is proposed based on a nonstationary image model where the image is assumed to be a nonstationary random field composed of a finite number of wide sense stationary (WSS) random fields distinguished by their covariance functions. Individual elements of the bank of filters are designed by optimal Wiener estimation based on the covariance matrix of each WSS random field. We introduce a decision rule based on the Modified Quadratic Discriminant Function (MQDF) [3], which is used to determine the best estimate at each pixel. It is shown that this decision rule requires less computational cost and storage than the maximum a-posteriori probability (MAP) decision rule [2] while offering similar performance.

Image Model and MAP Decision Rule

We consider the problem of estimating the image s(i,j) from the noisy observation x(i,j) given by x(i,j) = s(i,j) + n(i,j) where n(i,j) is a zero mean white noise process with known variance σ_n^2 and that is uncorrelated with s(i,j). Let S denote a k-dimensional column vector consisting of the lexicographic concatenation of the rows in the W x W square window centered at (i,j) of the original image. The vector equation for this model can be written as X = S + N where X and N are, respectively, the k-dimensional observation and noise vectors.

We assume that the original image is a nonstationary random field composed of M classes θ ($\theta = 1,2, \ldots, M$) of WSS random fields. Based on these assumptions, Ingle and Wood [1] modified the work of Lebedev and Mirkin [2] and derived an MAP decision rule which may be stated as follows: Select class θ, the class which exhibits the minimum discriminant function J, defined as

$$J = (1/2)\, X^T \Sigma_{X\theta}^{-1} X + (1/2) \ln |\Sigma_{X\theta}| - \ln p(\theta) \tag{1}$$

where $\Sigma_{X\theta} = \Sigma_{S\theta} + \Sigma_n$ where $\Sigma_{S\theta}$ is the covariance matrix corresponding to class θ and $\Sigma_n = \sigma_n^2 I$. This discriminant function is equivalent to the Quadratic Discriminant Function (QDF) commonly used in pattern recognition in the case of a zero mean vector for all the classes.

Modified Quadratic Discriminant Functions (MQDF)

The QDF is computationally expensive, requiring computation and storage of O(k^2) for k-dimensional feature vectors. This problem can be partially eliminated through the use of an MQDF. By using an eigenvector expansion of (1) and making the assumption that all small eignenvalues can be replaced by a small constant, Kimura, et al. [3] showed that a less compuationally intensive discriminant function with performance near that of the QDF may be expressed as:

$$J_2 = \frac{1}{h^2}\left[\|X\|^2 - \sum_{i=1}^{b} \left(1 - \frac{h^2}{\lambda_i}\right) \{\phi_i^T X\}^2 \right] + \ln h^{2(k-b)} \prod_{i=1}^{b} \lambda_i \tag{2}$$

It can be easily seen that this discriminant function, while still basically O(k^2), will require approximately b/k times the computation and storage of the QDF. Our restoration algorithm first classifies a pixel as belonging to one of M classes using the decision rule of (2) and then estimates the pixel intensity by using the appropriate Wiener filter for the selected class.

Experimental Results and Conclusion

A noisy image, degraded by additive white Gaussian noise of variance 225, was applied our algorithm with b values of 1, 5, 10, 15, 20, and 24. Figure 1(a-d) visually demonstrates the performance of the algorithm for various values of b. Figure 2 exhibits the Normalized Mean Square Error (NMSE) associated with the filters with different values of b. It is interesting to observe that the NMSE decreases only slightly with increasing b. Obviously, from (2), the computational complexity increases linearly with b. Table 1 summarizes the NMSE's and the computational complexity of the algorithms. Additionally, an image is shown in Figure 3 for performance comparison purposes resulting from the use of the best fixed Wiener filter from the bank of filters used in this investigation. Experimental results indicate that our MQDF-based algorithm can perform nearly as well as an MAP-based decision rule algorithm with substantially reduced computational cost.

REFERENCES

[1] V. K. Ingle and J. W. Woods, " Multiple model recursive estimation of images," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Washington, DC, pp. 642-645, Apr. 1979

[2] D. S. Lebedev and L. I. Mirkin, "Smoothing of two-dimensional images using the 'composite' model of a fragment," in *Iconics - Digital Holography - Image Processing*, USSR Inst. Inform. Transmission Problems, Academy of Sciences, pp. 57-62, 1975

[3] F. Kimura, K. Takashina, S. Tsuruoka and Y. Miyake, "Modified quadratic discriminant functions and the application to Chinese character recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, No. 1, pp. 149-153, Jan. 1987

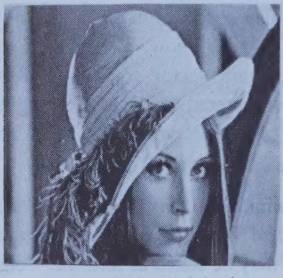


Fig. 1(a) Original

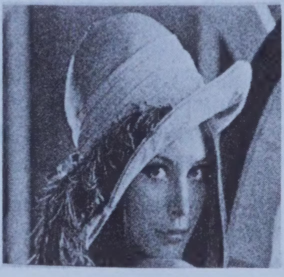


Fig. 1(b) Noisy

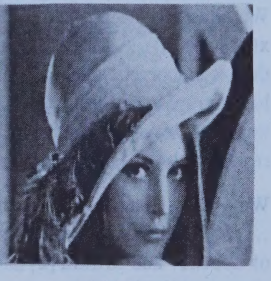


Fig. 1(c) b=1

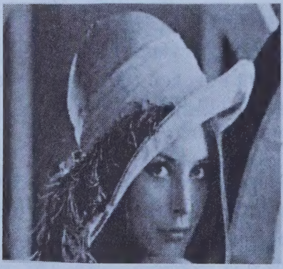
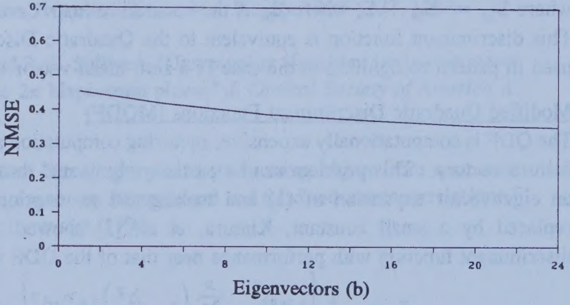


Fig. 1(d) b=24

Fig. 2 Algorithm Performance



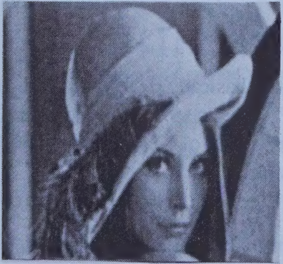


Fig. 3 Fixed Filter

TABLE 1

b	NMSE	Comp. Time
1	0.454	1
5	0.418	3.01
10	0.386	5.37
15	0.369	7.83
20	0.354	10.3
24	0.349	12.3
Fixed	0.664	0.21

LEAST SQUARES RESTORATION OF MULTI-CHANNEL IMAGES

Nikolas P. Galatsanos[1], *Aggelos K. Katsaggelos*[2], *Roland T. Chin*[3],

and *Allen Hillery*[3]

[1] Electrical and Computer Engineering
Illinois Institute of Technology
Chicago, Illinois 60616.

[2] Electrical Engineering and Computer Science
Northwestern University
Evanston, Illinois 60208.

[3] Electrical and Computer Engineering
University of Wisconsin
Madison, Wisconsin 53706.

SUMMARY

Multi-channel image restoration is a process to recover the set of ideal images observed by a multisensor imaging systems, such as color and multispectral satellite scanning imagery. Most of the work in restoration has concentrated in the restoration of single channels, failing to utilize the interrelated information between channels. Hunt and Kubler [1] presented a multi-channel minimum mean square error filter (MMSE) based on the assumption that the signal autocorrelation, describing the between channel (or spectral) and the within channel (or spatial) relationship, is separable. In [2], and [4] we presented multi-channel MMSE filters that do not utilize the separability assumption. A significant amount of prior knowledge is required for multi-channel MMSE filters. Although the filters are optimally derived, their true success in restoring real-world images depends on the accuracy of statistical knowledge about the signal and the noise. To be more specific, the MMSE filters require accurate estimation of the image and noise power spectra. In this paper we consider the problem of multi-channel restoration using *both within and between-channel deterministic information.* This alleviates some of the problems associated with the MMSE filter.

Multi-channel least squares restoration filters are developed using two approaches, the set theoretic and the constrained optimization ones. For the set theoretic approach which has been applied to independent channel restoration [3], a priori knowledge restricts the solution $\mathbf{f}$ to lie in the following ellipsoids

$$\mathbf{f} \in S_{\mathbf{f}/g_i} = [\, \mathbf{f} : \, \|\overline{H}_i \mathbf{f} - g_i\|^2 \le \varepsilon_i^2 \,], \tag{1}$$

and

$$\mathbf{f} \in S_{\mathbf{f}} = [\, \mathbf{f} : \, \|\mathbf{Q}\,\mathbf{f}\|^2 \le E^2 \,], \tag{2}$$

where

$$\overline{H}_i = [\, H_{i1}, H_{i2}, \ldots, H_{iN} \,], \tag{3}$$

with H_{ij} the degradation between channels i,j and $\mathbf{Q}$ the regularization operator that captures both within and between channel a priori information about the signal $\mathbf{f}$ and renders a better condition number to the multi-channel blur $\mathbf{H}$. Thus $\mathbf{f}$ must also lie in their intersection. The center of a bounding ellipsoid of the intersection is used as the estimate $\mathbf{f}$. Such an $\mathbf{f}$ is given by

$$[\sum_{i=1}^{N} \lambda_i \overline{H}_i{}^t \overline{H}_i + Q^t Q]\,\hat{\mathbf{f}} = \sum_{i=1}^{N} \lambda_i \overline{H}_i{}^t g_i \tag{4}$$

where $\lambda_i = (E/\varepsilon_i)^2$. If the parameter E is not available, then an $\hat{\mathbf{f}}$ is sought which

$$\textit{Minimizes} \quad ||\mathbf{Q}\,\mathbf{f}||^2 \tag{5}$$

$$\textit{subject} \ ||\overline{H}_i\,\mathbf{f} - g_i||^2 = ||n_i||^2 = \varepsilon_i^2, \text{ for } i=1,\ 2\ ,.,\ N. \tag{6}$$

Using Langrange multipliers it can be shown that the solution $\hat{\mathbf{f}}$ to this constrained least squares problem is also given by Equation (4). However, in this case the coefficients λ_i are Langrange multipliers and must be selected such that $\mathbf{f}$ satisfies Equations (6). A Newton method has been employed towards this task. The Jacobian of Equation (6) was analytically evaluated. We have shown using a geometric interpretation for the estimates of both filters, that the set theoretic approach yields a sharper but noisier estimate than the constrained optimization one.

Color images, that is three-channel imagery with red, green, and blue components, are considered as representative examples of multi-channel images. Constraints that capture the within and between-channel properties of color images are developed in the form of multi-channel regularizing operators $\mathbf{Q}$. Three-dimensional ($3D$) weighted regularizing operators are proposed based on the ratios of the norms of each channel. Issues associated with the computation of the two estimates are addressed. The computational framework developed in [2] is used for the solution of Equation (4) and for the implementation of the Newton iterations for the constrained optimization approach. Experiments with color images are shown, where: (a) We verified that utilizing between channel information yields better restorations. (b) We confirmed that the set theoretic approach yields sharper but noisier estimates that the constrained optimization one. (c) We tested a number of regularizing operators and we verified that the $3D$ weighted regularizing operators yield better restorations.

REFERENCES

[1] B. R. Hunt and O. Kubler, "Karhunen-Loeve multispectral image restoration. Part I: Theory," *IEEE Trans. Acoustic, Speech, and Signal Processing,* vol. ASSP-32, no. 3, pp. 592-600, June 1984.

[2] N. P. Galatsanos and R. T. Chin, "Digital restoration of multi-channel images," *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. ASSP-37, No. 3, pp. 415-421, March 1989.

[3] A. K. Katsaggelos, J. Biemond, R. W. Schafer, and R. M. Mersereau "A regularized iterative image restoration algorithm", *IEEE Trans. Acoustic, Speech, and Signal Processing,* to appear, April 1991.

[4] N. P. Galatsanos and R. T. Chin, "Restoration of Color Images by Multi-channel Kalman filtering," *IEEE Trans. Acoustics, Speech, and Signal Processing,* to appear.

TEXTURE SEGMENTATION BASED ON RANDOM FIELD MODELS AND KNOWLEDGE BASED REASONING

Hong Li and R. L. Kashyap
Purdue University
School of Electrical Engineering
West Lafayette, IN 47907

1. INTRODUCTION

Segmentation of a given image is an important task in image processing and computer vision. Recently more attention has been paid to segmentation based on random field models and reasonable progress have been demonstrated[1-6]. Some results have been obtained by researchers using expert systems in segmentation[7]. In this paper, texture segmentation is viewed as a transformation of our prior knowledge about texture boundaries into a boundary map. Prior knowledge is classified into local prior knowledge and global prior knowledge. The former is first used for segmentation based on random field models and the latter is used through knowledge based reasoning. A model and knowledge based segmentation algorithm is thus formed from these two parts. The proposed algorithm is tested on various real textures and scenes and its efficiency is demonstrated.

2. MODEL BASED SEGMENTATION ALGORITHM

For each pixel in the image, one vertical strip with 10 rows and 4 columns and a horizontal strip with 4 rows and 10 columns are constructed with this pixel at the center. Based on the local prior knowledge, it is assumed that the two strips for each pixel have at most one boundary with six possible directions. The two strips are fitted to NSAR models for six possible boundary directions. Then six likelihood values corresponding to six boundary directions are obtained. The maximum value is then selected as the likelihood value for this pixel. The corresponding direction with this likelihood value is chosen as the direction of the possible boundary centered at this pixel. Thus each pixel in the input image is assigned a likelihood value and a boundary direction. The likelihood and direction values from all the pixels in the image form two new feature images, called the likelihood image and the direction image respectively. To these two feature images, a new spatial filtering scheme is formed and the local prior knowledge is used efficiently. A new model based segmentation algorithm is proposed.

3. KNOWLEDGE BASED SEGMENTATION ALGORITHM

The boundary map obtained by the local prior knowledge contains mistakes due to lack of the global prior knowledge. Some boundaries are falsely detected, some have wrong directions, and some true boundaries are lost. By our global prior knowledge, it is assumed that the texture boundaries are continuous and closed. Then a set of rules for knowledge based reasoning are constructed based on the global prior knowledge. Using these rules, the reasoning algorithm deletes falsely detected boundaries, corrects wrong boundary directions, and links lost boundaries. Thus a knowledge based reasoning algorithm is provided for improving the boundary map given by the model based algorithm.

4. EXPERIMENTS

Different kinds of real textures and real scenes are employed to test the model and knowledge based algorithm. Three testing images are chosen in figure 1. In figure 1(a), the image is composed of four textures - grass(top-left), woodgrain(top-right), water(bottom-left), and leather(bottom-right). The image in figure 1(b) is formed from five textures - leather (top-horizontal - strip), water (middle - horizontal - strip), grass (bottom - horizontal - strip and right - circle), woodgrain (middle - vertical - strip), and straw (left - circle). A aerial picture of a bay area is given in figure 1(c). The segmentation results by the model based algorithm are displayed in figure 2(a - c) respectively. In figure 3(a - c), the results by the knowledge based reasoning algorithm are indicated.

5. CONCLUSIONS

In this paper a new segmentation algorithm, which combine a model based algorithm and a knowledge based algorithm, is proposed. The model based algorithm is implemented on the likelihood image and the direction image. Better results are obtained than those on the pixel images directly. Many mistakes made by the pure model based algorithm can be reduced efficiently by applying the knowledge based algorithm. The combination of the two algorithms increases the power of the segmentation algorithm significantly. Different textured images are used to test the new algorithm and the excellent segmentation results of the new algorithm are demonstrated.

REFERENCES

[1] R.L. Kashyap and K.-B. Eom, "Texture Boundary Detection Based on the Long Correction Model," IEEE Trans. PAMI, vol.11, January 1989.

[2] K.-B. Eom and R.L. Kashyap, "Composite Edge Detection with Random Field Models," IEEE Trans. SMC, vol.20, January 1990.

[3] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images," IEEE Trans. PAMI, vol.6, November 1984.

[4] H. Derin and H. Elliott, "Modeling and Segmentation of Noisy and Textured Images Using Gibbs Random Fields," IEEE Trans. PAMI, vol.9, January 1987.

[5] S. Lakshmanan and H. Derin, "Simultaneous Parameter Estimation and segmentation of Gibbs Random Fields Using Simulated Annealing," IEEE Trans. PAMI, vol.11, August 1989.

[6] D. Geman and S. Geman, C. Graffigne, and P. Dong, "Boundary Detection by Constrained Optimization," IEEE Trans. PAMI, vol.12, July 1990.

[7] A.M. Nazif and M.D. Levin, "Low Level Image Segmentation: An Expert System," IEEE Trans. PAMI, vol.6, September 1984.

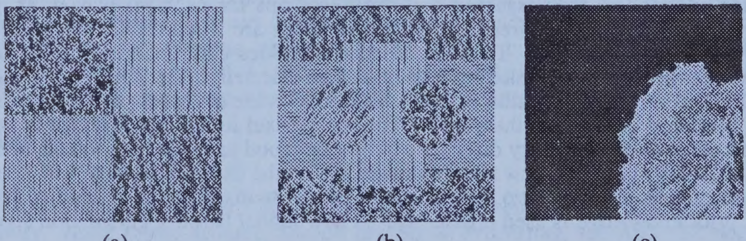

(a) (b) (c)

Figure 1. Three images to be segmented

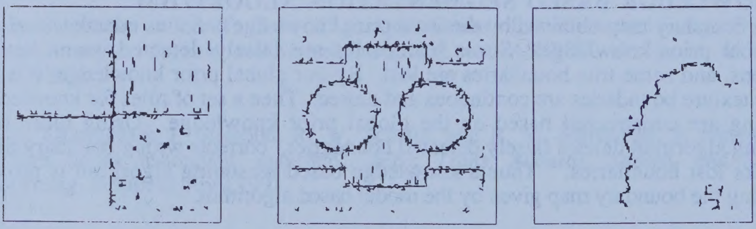

(a) (b) (c)

Figure 2. The boundary map given by the model based algorithm

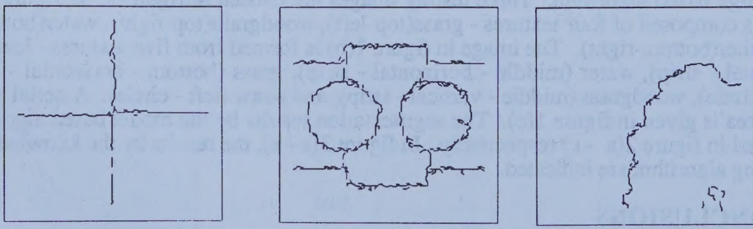

(a) (b) (c)

Figure 3. The boundary map produced by the model and knowledge based algorithm

Rotation Invariant Object Detection Using ANOVA-Based Data Model

M.Hafed Benteftifa and Ludwik Kurz

Dept of Electrical Engineering
Polytechnic University, 333 Jay Street
Brooklyn, New York 11201, USA

Abstract

The detection of two-dimensional objects embedded in a noisy environment subject to arbitrary rotations is addressed in this paper. The procedure is a two-stage process: a learning and a detecting stage. During the learning stage, the data is transformed using a complex conformal mapping which results in a rotation invariant representation. The transformed template is then used to represent the objects from which features are extracted. An existing procedure that operates well in a noisy environment is used during the detection stage. Results of computer simulations show the effectiveness of the new procedure in detecting objects irrespective of existing rotations between test and reference objects.

I. Introduction

In recent papers [2]-[3] we introduced techniques for detecting objects in two-dimensional scenes using ANOVA techniques. The features used in the recognition process are based on *F*-statistics and contrast functions. A prime objective was to achieve reliable detection in the presence of noise. A major assumption in the detection process is that the test object must be properly aligned with respect to the reference object. In other words, the contrast calculation, in the case of the contrast-based algorithm [2], is performed according to the orientation of the object, resulting in a different problem when there is a rotation between test and reference objects. A generalization of the contrast-based algorithm would of course include this rotation in the contrast calculation regardless of any rotation angle that might exist. In the literature, this is referred to as the 2-D rotation-invariant object recognition problem. Considerable attention has been devoted to it due to its numerous applications, especially in medecine, machine vision and target extraction.

Moment invariants have been used in solving this problem by many researchers [4], [5] and [6] . They rely on algebraic forms that are invariant to scale and rotation transformation of data. A set of seven moments is usually derived for the test object and is compared to those corresponding to the reference object. Relevant results can be found in [5]. In this paper, we use a complex conformal mapping on the data before any detection procedure that we developed previously in [2] and [3] is actually used.

Complex conformal mapping has its foundation in the biological behavior of the human visual system. Several studies have been conducted to understand the transformation occuring between the retinal and the cortical planes [9]. A viable mathematical model has been shown to be equivalent to a logarithmic mapping between the retinal plane of the eye and the cortex of the brain. Procedures based on this philosophy have been devised and used in the detection of two-dimensional objects irrespective of rotation and scaling of the data [7]-[8]. The present study makes use of the complex mapping in the early processing stage. As a result, the original data is transformed in such a way as to remove the effect of an existing rotation. We deliberately concentrate on the rotational invariance since we believe that it is more serious than the scaling effect. Needless to say, the scaling problem can be handled in the same manner. This paper is organized as follows.

In the first section, we review the complex conformal mapping as applied to invariant representation of the data. Then, an efficient detection technique is developed using an existing detection procedure in conjunction with the mapping of the data.

Finally, we present experimental results that demonstrate the effectiveness of the present methodology in detecting objects regardless of any existing rotation.

II. Invariant Representation

2.1 Complex conformal mapping scheme

The detection procedures in [2] and [3] use a rectangular representation of data, i.e., the location of each pixel is ascertained in terms of the vertical and horizontal coordinate pair (i,j) i=1,..., m ; j=1,..., n. This representation,i.e., sampling, is inadequate if one is to take into account any possible rotation of the object. Therefore, an alternative representation in terms of polar coordinates is useful for this case. Any point in the plane is thus referenced by the pair (ρ, θ), where ρ is the radial distance from an appropriately chosen origin and θ is the polar angle. An invariant representation is then possible by using what is commonly referred to as the complex conformal mapping. In order to understand the mechanisms behind the mapping, we first review some useful definitions.

Consider a point in the observation space which is referenced by

$$z = \rho\, e^{j\theta} \tag{1}$$

Now, consider a logarithmic transformation of (1), obtaining

$$w = \ln (z) = \ln \rho + j\,\theta \tag{2}$$

which can also be expressed as

$$w = \mathbf{u} + j\,\mathbf{v} \tag{3}$$

where $\mathbf{u} = \ln \rho$ and $\mathbf{v} = \theta$.

In terms of the rectangular coordinates (i,j) the pair ($\mathbf{u}$,$\mathbf{v}$) can also be expressed as

$$\begin{aligned} \mathbf{u} &= \ln \left(i^2 + j^2\right)^{1/2} \\ \mathbf{v} &= \tan^{-1}\left(\frac{j}{i}\right) \end{aligned} \tag{4}$$

Based on an analogy with the human visual system (HVS), the transformation is commonly referred to as the retino-cortical mapping between retinal plane, represented in the present case by the polar plane (ρ,θ), and the cortical plane of the brain represented by the cartesian plane ($\mathbf{u}$,$\mathbf{v}$). Interesting properties that are associated with this particular mapping are easily obtained from (1) and (2). In the following, the image in the polar plane is referred to as the retinal image and its transformed version as the cortical image.

a. Rotation Invariance.

Suppose that the retinal image represented by $\mathbf{Y}(\rho,\theta)$ is rotated by an angle θ_0. The corresponding pair ($\mathbf{u}'$,$\mathbf{v}'$) is obtained by first considering

$$z' = \rho\, e^{j(\theta+\theta_0)} \tag{5}$$

and, using the logarithmic transformation, one obtains

$$w' = \ln\ \rho + j(\theta+\theta_0) \tag{6}$$

and hence

$$\begin{aligned} \mathbf{u}' &= \ln\ \rho \\ \mathbf{v}' &= \theta+\theta_0 \end{aligned} \tag{7}$$

As a result, a rotation in the retinal plane yields a shift along the v-axis in the cortical plane.

b. Scale Invariance

Suppose the image $\mathbf{Y}(\rho,\theta)$ is scaled by a factor *a*. Then, the corresponding point becomes

$$z'' = a\,\rho\, e^{j\theta} \tag{8}$$

which is transformed to the pair

$$\begin{aligned} u'' &= \ln a\rho = \ln a + \ln \rho \\ v'' &= \theta \end{aligned} \tag{9}$$

The scaling thus results in a shift by $\ln(a)$ along the u-axis.

These simple properties have been used by Massone *et al* [7] in the detection of objects in the case of binary images. An extension to the general case of grey scale images has been shown to be possible [8].

The sampling of the retinal image is performed by using a non-uniform sampling grid which is formed by exponentially spaced concentric circles. The sampling is accompanied by a reduction in the image size. The number of concentric circles yields the maximum size of the u-axis while the number of points on the circle determines the maximum size of the v-axis. The exact determination of (u_{max}, v_{max}) is based on such factors as the object size. The reader is referred to [7] which provides a detailed description of the procedure involved in the selection of the pair (u_{max}, v_{max}) for both the single and multi-object cases.

2.2 Detection procedure

The present algorithm is a two-stage procedure: a learning or training stage and a detection stage. The training stage involves the extraction of useful features necessary for the detection while the detection stage consists of comparisons of the test features to the reference features.

The first step during the learning phase is the mapping of the collected data onto the cortical plane. As a result, any existing rotation is reduced to a translation along the v-axis. The object to be detected is represented by the template $\mathbf{RT}(i,j)$ where $i=1,\ldots,m;\ j=1,\ldots,n$. The template obtained after sampling and mapping using (1) and (2) , denoted by $\mathbf{CT}(u,v)$, is then stored as the new object template.

In the binary image case, Massone *et al* [7] also computed the orientation of the principal axis of inertia as well as the position of the center of gravity in the cortical plane. The usefulness of these features stems from the fact that the origin of the v-axis is shifted to coincide with the orientation of the principal axis of inertia while the center of gravity becomes the new origin. Their calculation in the noisy image case is, however, highly susceptible to the effect of the corrupting noise. In the present case, the shift along the v-axis is detected by considering the projection of the cortical image on a cylinder. Hence, the v-axis can be thought of as being wrapped around the cylinder. Thus, a cyclic shift of the image will eventually recover the position of the image with respect to the shift.

Next, features for the detection are extracted . There are two possible procedures that are available to us: the contrast-based procedure [2] and the transformation-based procedure [3] . The choice of either one is rather subjective, and the decision which one to use is based on such factors as the size and shape of **CT**(u,v). In this paper, we use the transformation-based procedure.

2.2.1 Transformation-based procedure

Originally developed by Chang and Kurz [11] for the independent data case and extended to include dependency among data by Benteftifa and Kurz [3]-[12], the present procedure is based on the transformation of **CT**(u,v) into three arrays: background array **B**(u,v), target array **T**(u,v) and the combination of both which is referred to as the global array **S**(u,v). The observations within each array are parameterized by the following two-way ANOVA model

$$\Omega: \begin{cases} y_{uv} = \mu + \alpha_u + \beta_v + e_{uv} \quad u=1,2,\dots, u_1;\ v=1,2,\dots, v_1 \\ \sum\limits_{v=1}^{v_1} \beta_v = 0 \,;\ \sum\limits_{u=1}^{u_1} \alpha_u = 0 \\ (e_{uv})\,\text{independent}\,N(0, \sigma^2) \end{cases} \tag{10}$$

where μ is the general mean, β_v are the column effects and α_u represent row effects within the array.

Starting with a minimal background, the learning stage is initiated by transforming **CT**(u,v) to obtain the three arrays of concern. At the final stage, **B**(u,v) and **T**(u,v) should be homogeneous,i.e., contain only background and target pixels, respectively. On the other hand, **S**(u,v) should be heterogeneous since it includes both types. These conditions are tested by forming column and row statistics corresponding to each array and testing them against tabulated threshold for a given test level α. The detection conditions are summarized in the following [3].

1- **S** is heterogeneous, i.e., FSc > F_α and FSr < F_α
2- **B** is homogeneous, i.e., FBc < F_α and FBr < F_α (11)
3- **T** is homogeneous, i.e., FTc < F_α and FTr < F_α
4- Contrast exists between columns k and k+1

The tabulated thresholds F_α set depends on the dimensions of **T**, **B** and **S**.

If all conditions in (11) are satisfied, the learning stage is successfully terminated by recording the dimensions of **B**(u,v), **T**(u,v) and **S**(u,v) as well as the transformation rules.

The detection process is then initiated where the test template is transformed to yield target, background and global arrays. Using (11) with the same thresholds, the object is declared to be present if all the conditions are satisfied.

2.2.2 Rotation invariant detection procedure

At this stage, the information from the cortical image **CT**(u,v) has been transformed to a set of detection features characterizing the original object **RT**(i,j). Note that if the image contains multiple objects, the above learning phase is to be repeated for each object.

Before we embark on the description of the detection stage for the rotation invariant procedure, we review the possible scanning scheme of the image with unknown objects. First, we consider a row scanning with a mask having the same size as the object template. Since at each scan it is required to perform an additional cyclic scanning to detect an eventual shift, it would appear that it requires a high computation time. An alternate approach calls for an initial segmentation of the image so as to separate the various objects. Hence, it would seem to be advantageous to use a segmentation procedure in conjunction with a cyclic scanning.

The detection phase is then initiated with the following hypothesis-alternative pair

H: $F=F_N$

K: $F \neq F_N$

where F_N is the reference set of features corresponding to the transformation-based procedure and F is the test feature set.

Assume that the image has already been segmented, resulting in the objects O_1, O_2, ..., O_n. Each object is then mapped using (1) and (2). Using a mask with the same dimensions as the cortical reference object, a cyclic scanning is initiated. At each pass, the feature set, i.e., FO_i i=1,...,n, is extracted and compared to the reference set F_{Ni}. If the set satisfies the detection condition in (11), the object is detected and its position recorded. Note that we also obtain the corresponding rotation angle existing between the test and reference objects.

III. Experimental Results

In order to assess the performance of the proposed procedure , several experiments were conducted. The primary objective of the simulations was to test the procedures when presented with rotated versions of the target.

An "I" shaped object was selected as the reference object. The image containing the object is of size 256x256. The object covers an area of about 42x42 pixels. Two different levels of noise were chosen, namely, $\sigma = 1.00$ and $\sigma = 2.21$ resulting in SNR levels of 16.9

db and 10 dB. The initial step is the complex mapping of the image. The cortical image is of dimensions 64x64. Hence, the samples are taken at 5.62 degrees apart on exponentially spaced circles. The number of circles is in this case equal to 64.

The new representation in the cortical plane is used as the template for the transformation-based procedure described in [3]. The learning stage resulted in the arrays **B, T** and **S** having dimensions 16x65, 16x191 and 16x256, respectively. Note that the ratio of target-to-background pixels is close to 3. Two rotated versions of the image are used as test images. The angles of rotation are 90^{o} and 135^{o}. For each case, a cyclic scanning is initiated starting from the origin in the cortical plane, which is arbitrarily fixed as the upper left hand corner. The final results are sorted by using the column statistic corresponding to the array **S**. Figures 4-5 show the results obtained under the various noise conditions. Note that in each case, there are two peaks, separated by 180^{o}, corresponding to the possible location of the object. This ambiguity is due to the nature of the transformation itself. The estimates of the rotation angle are shown in table 1.

Rotation angle	Peak 1	Peak 2
90^{o}	89.92^{o}	269.76^{o}
135^{o}	134.88^{o}	314.72^{o}

Table 1: Angle estimates.

As we see, the estimate is very close to the actual values of the rotation angle, for the chosen value of the sampling, and for the two values of σ.

IV. Conclusions

In this paper, we addressed the problem of detecting objects embedded in a noisy environment regardless of any existing rotation between test and reference versions of the object. We used a conformal complex mapping under which a rotation is reduced to a translation. Once the invariant representation was acquired, a transformation-based detection procedure was used . The input template in this case is the complex mapped version of the reference object. Using cyclic scanning, we were able to detect the object as well as estimate the angle by which the test object is rotated with respect to the reference. The estimate of the angle was found to be very close to the actual value and this was true for various values of the corrupting noise. The results of computer simulations show the effectiveness of the procedure.

References

[1] H.Scheffe, *The Analysis of Variance*, John Wiley, 1959.

[2] M.H Benteftifa, L.Kurz," ANOVA-Based Linear Contrast Techniques in Object Detection", *Submitted to Journal Optical Society of America, Part A.*

[3] M.H Benteftifa, L.Kurz, " Two-Dimensional Object Detection in Correlated Noise",*Submitted to Pattern Recognition.*

[4] M.K.Hu, "Visual Pattern Recognition by Moments Invariants", *IRE Trans. Inf. Theory*, IT-8, 179-187, 1962.

[5] M.R Teague," Image Analysis Via the General Theory of Moments," *Journal Optical Society of America*, vol 70, no8, pp920-930, August 80.

[6] Y-S Abu Mostafa and D.Psaltis, "Recognitive Aspects of Moment Invariants," *IEEE Trans on Pattern Analysis and Machine Intelligence*, PAMI-6, 698-706, 1984.

[7] L.Massone, G.Sandini and V.Tagliasco," Form-Invariant Topological Mapping Strategy for 2D Shape Recognition," *Comput Vision, Graphics Image Processing*, vol.30, pp.169-188, 1985.

[8] R.A Messner and H.H.Szu, " An Image Processing Architecture for Real-Time Generation of Scale and Rotation Invariant Patterns," *Comput Vision, Graphics Image Processing*, vol.31, pp.50-66, 1985.

[9] E.L.Schwartz, " Spatial Mapping in the Primate Sensory Projection:Analytic Structure and Relevance to Perception," *Biol.Cybern*, vol.25, pp.181-194, 1977.

[10] C.H.Teh and R.Chin,"On Image Analysis by the Methods of Moments," *IEEE Trans on Pattern Analysis and Machine Intelligence*, vol.10, no 4, pp. 496-513, July 1988.

[11] E.H.Chang, L.Kurz," Object Detection and Experimental Designs", *Computer vision, Graphics and Image Processing*, 40, pp 147-168, 1987.

[12] M.H Benteftifa, L.Kurz, " An Approach to Object Detection in Correlated Noise", *23rd Annual Conference on Information Sciences and Systems*, The John Hopkins University, pp 217-221, March 1989.

[13] M.H Benteftifa, L.Kurz, " Linear Contrast Techniques in Detecting Objects Corrupted by Noise", *International Conference on Image Processing, ICIP'89*, Singapore, September 1989.

[14] C.F.R.Weiman and G.Chaikin," Logarithmic Spatial Grids for Image Processing and Display," *Comput . Graphics Image Processing*, vol.11, pp. 197-226, 1979.

[15] M.G Kendall, *The Advanced Theory of Statistics, vol 1*. Griffin Ltd. 1958.

[16] M.H Benteftifa, *Object Detection Via Linear Contrast Techniques*, PhD Dissertation, Polytechnic University, New York, June 1990.

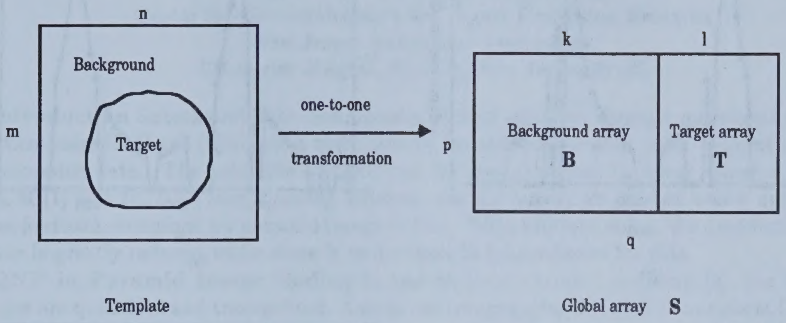

n
Background
Target
m
one-to-one
transformation
k
l
Background array
B
Target array
T
p
q
Template
Global array S

Fig. 1: Transformation procedure.

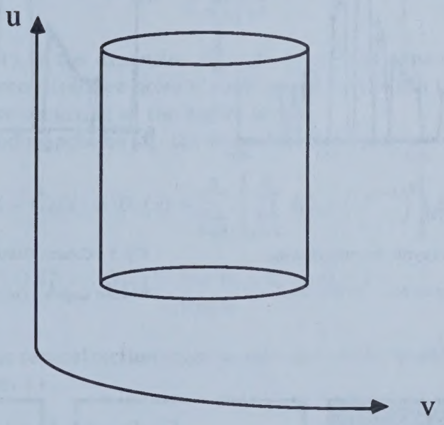


Fig. 2: Cyclic scanning.

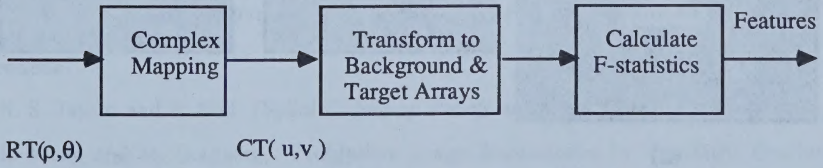
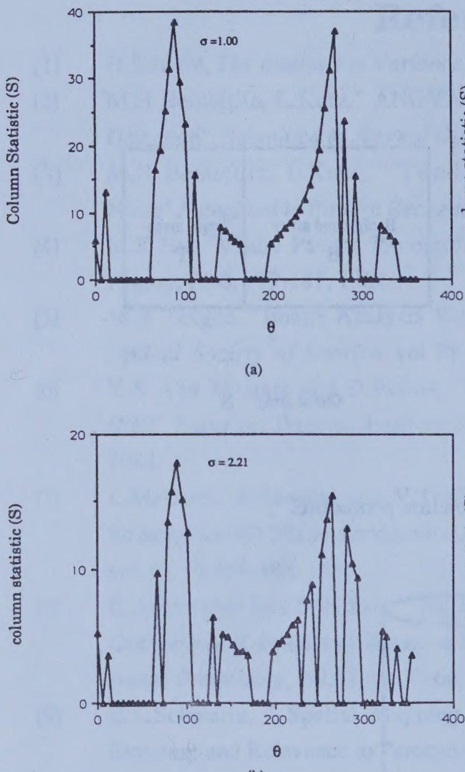

Complex Mapping
Transform to Background & Target Arrays
Calculate F-statistics
Features
RT(ρ,θ)
CT(u,v)

Fig. 3: Learning/Detection stage.



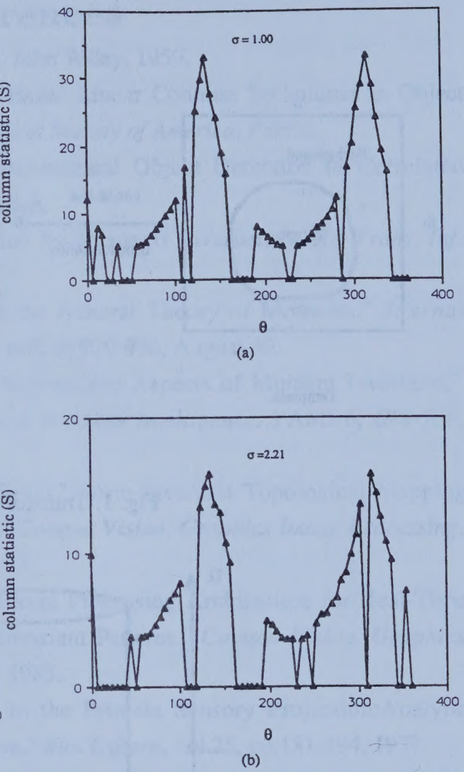


Fig. 4: Column Statistic vs cyclic scanning position. rotation angle = 90°. (a) $\sigma = 1.00$. (b) $\sigma = 2.21$.

Fig. 5 : Column Statistic vs cyclic scanning position. Rotation angle = 135°.(a) $\sigma = 1.00$. (b) $\sigma = 2.21$.

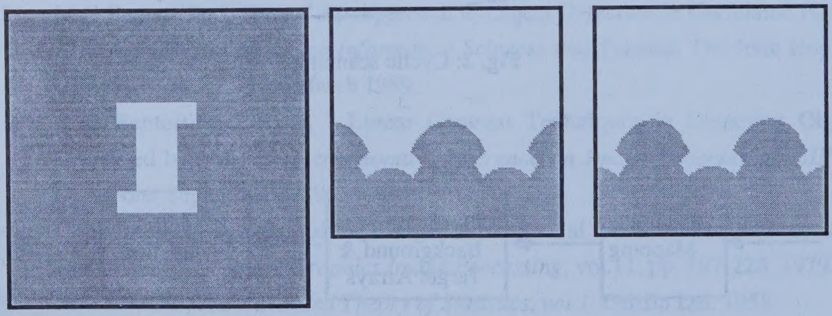

(a) (b) (c)

Fig. 6: Conformal mapping of test image "I". (a) original image in retinal plane. (b) image in cortical plane $\theta = 0^{\circ}$. (c) image in cortical plane $\theta = 90^{\circ}$.

A New QNF for Pyramid Image Coding[1]

Chung H. Lu, Zhili Lin

Department of Electrical Engineering
Center for Communications and Signal Processing Research
New Jersey Institute of Technology
University Heights, Newark, New Jersey 07102

1 Introduction Substantial data compression is often achieved through quantization. Quantization noise feedback (QNF) is a technique to reduce quantization noise without increasing transmission rate. The principle of QNF can be found applied in many systems including DPCM [1] and DCT [2] image coding schemes. In this paper, we present a new quantization noise feedback technique for pyramid image coding. With this technique, the quantization noise power is greatly reduced while there is no increase in transmission bit rate.

2 QNF in Pyramid Image Coding In conventional pyramid encoding [3], the Laplacian images are quantized and transmitted. Assume an image $g_k(n_1, n_2)$ with Z-transform $G_k(z_1, z_2)$. Let H_k^r denote the reduction filter R, ρ $(0 < \rho < 1)$ the reduction ratio. Then $G_1(z_1, z_2) = H_0^r(z_1^\rho, z_2^\rho) G_0^r(z_1^\rho, z_2^\rho)$. For simplicity, we replace z_1^ρ, z_2^ρ with z^ρ. The reconstruction error is given by

$$E_{\mathrm{CON}}(z) \triangleq G_0(z) - \hat{G}_0(z) = Q_0(z) + \sum_{k=1}^{n} \left\{ \prod_{j=0}^{k} H_{j-1}^e\left(z^{\eta^{j-1}}\right) \right\} Q_k\left(z^{\eta^k}\right)$$

where $H_k^e(z)$, $(H_{-1}^e(\cdot) \triangleq 1)$, is the expansion filter E, $\eta = \frac{1}{\rho}$ the expansion ratio, and Q_k the quantization error. The reconstruction noise is contributed by the 0th layer quantization noise plus the quantization noise occurring at the higher layers.

For QNF I pyramid coding scheme [4], the reconstruction error is

$$E_{\mathrm{QNF\ I}}(z) \triangleq G_0(z) - \hat{G}_0(z) = D_0(z) + \sum_{k=1}^{n} \left\{ \prod_{j=0}^{k} H_{j-1}^e\left(z^{\eta^{j-1}}\right) \right\} D_k\left(z^{\eta^k}\right)$$

$$D_k(z) = \begin{cases} \{1 - H_k^e(z) H_k^q(z)\} Q_k(z) & k = 0, 1, 2, \cdots, n-1 \\ Q_n(z) & k = n \end{cases}$$

With QNF II (Figure), the reconstruction error is only due to the quantization error occurring at the reconstruction layer, *i.e.*,

$$E_{\mathrm{QNF\ II}}(z) \triangleq G_0(z) - \hat{G}_0(z) = Q_0(z)$$

Apparently, $E_{\mathrm{QNF\ I}}(z)$ is smaller than $E_{\mathrm{CON}}(z)$ by an amount depending on the reduction and expansion filters. $E_{\mathrm{QNF\ II}}(z)$ is arguably the smallest.

3 Simulation Results Tables 1, 2 and 3 show SNR and BPP data for the conventional pyramid, QNF I pyramid and QNF II pyramid respectively. Generally, QNF I performs slightly better than the conventional pyramid and QNF II is significantly superior to QNF I and the conventional pyramids.

References

[1] N. S. Jayant and P. Noll, *Digital Coding of Waveform*, Prentice-Hall, 1984.

[2] L. Wang and M. Goldberg, "Progressive Image Transmission by Transform Coefficient Residual Error Quantization," IEEE Trans. Commun., Vol. 36, No. 1, pp.75-85, Jan. 1988.

---

[1] The work reported in this paper is supported in part by an SBR grant from NJIT Research Office.

[3] P. J. Burt and E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code," IEEE Trans. Commun., Vol COM-31, No. 4, pp. 532-540, April 1983.

[4] Chung H. Lu and Jack Wang, "Quantization Noise Feedback in Pyramid Coding", *1990 Picture Coding Symposium*, p. 12.5, Cambridge, Mass., March 1990.

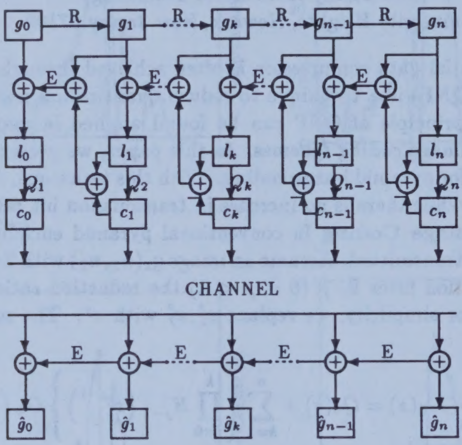

g_0
R
g_1
R
g_k
R
g_{n-1}
R
g_n
E
E
E
E
l_0
l_1
l_k
l_{n-1}
l_n
Q_1
Q_2
Q_k
Q_{n-1}
Q_n
c_0
c_1
c_k
c_{n-1}
c_n
CHANNEL
E
E
E
E
$\hat{g}_0$
$\hat{g}_1$
$\hat{g}_k$
$\hat{g}_{n-1}$
$\hat{g}_n$

Figure 1: Pyramid With QNF II

Table I. Conventional Pyramid

Pyramid Layer	Quantization Level	3	9	15	21	27
2	SNR	22.84	30.73	32.70	34.02	34.78
	BPP	1.94	3.58	4.46	4.86	5.13
3	SNR	22.22	28.37	29.91	31.00	31.56
	BPP	2.03	3.68	4.60	4.97	5.21
4	SNR	21.54	26.42	27.90	28.81	29.04
	BPP	2.04	3.67	4.59	4.96	5.19

Table II. Pyramid With QNF I

Pyramid Layer	Quantization Level	3	9	15	21	27
2	SNR	22.92	30.81	32.79	34.09	34.79
	BPP	1.94	3.58	4.46	4.48	5.13
3	SNR	22.52	28.40	29.99	31.05	31.60
	BPP	2.03	3.68	4.59	4.97	5.21
4	SNR	21.85	26.50	27.98	28.95	29.34
	BPP	2.04	3.67	4.58	4.95	5.18

Table III. Pyramid With QNF II

Pyramid Layer	Quantization Level	3	9	15	21	27
2	SNR	29.80	34.20	34.97	35.48	36.36
	BPP	1.98	3.95	4.68	5.05	5.29
3	SNR	28.79	30.54	31.75	32.52	33.31
	BPP	2.07	3.91	4.77	5.11	5.34
4	SNR	26.31	28.53	29.89	30.68	31.64
	BPP	2.08	3.76	4.67	5.07	5.29

Adaptive Edge Detection Based Nested Median Filter

Humor Hwang and R. A. Haddad
Electrical Engineering and Computer Science
Polytechnic University
Brooklyn, New York 11201

Abstract

To reduce edge shifting due to impulsive noise near a signal edge while suppressing impulsive noise as well as nonimpulsive noise, we propose an edge detection based median-type filter which combines properties of both linear and median filters.

Our filter consists of two levels. The first level is basically an optimal edge detector based on signal and noise models. For heavy-tailed noise(e.g., Laplacian noise), the maximum likelihood estimators for signal levels at either side of an edge are a pair of left and right median filters. The difference of these median filter outputs are then compared with a threshold in a hypothesis test which detects the presence of an edge. If an edge is detected, the second level algorithm is a 3-point median filter processing the left median and the right median outputs of the first level and the test point itself. This is called a *nested median filter*(NMF). If the first level asserts there is no edge, the second level algorithm is simply the weighted average of the left and right median filter outputs. The latter is called an *median averaging filter*(MAF).

If the noise is short-tailed(e.g., Gaussian), or middle-tailed(e.g., uniform), the first level median filters are replaced by linear mean filters and max-min filters, respectively.

We show that at the noisy edge point the NMF is substantially superior both theoretically and experimentally to the median filter, the $\alpha-$TM filter, and the STM filter[1] in two respects : (a) output mean error and (b) output root mean square error.

1. Introduction

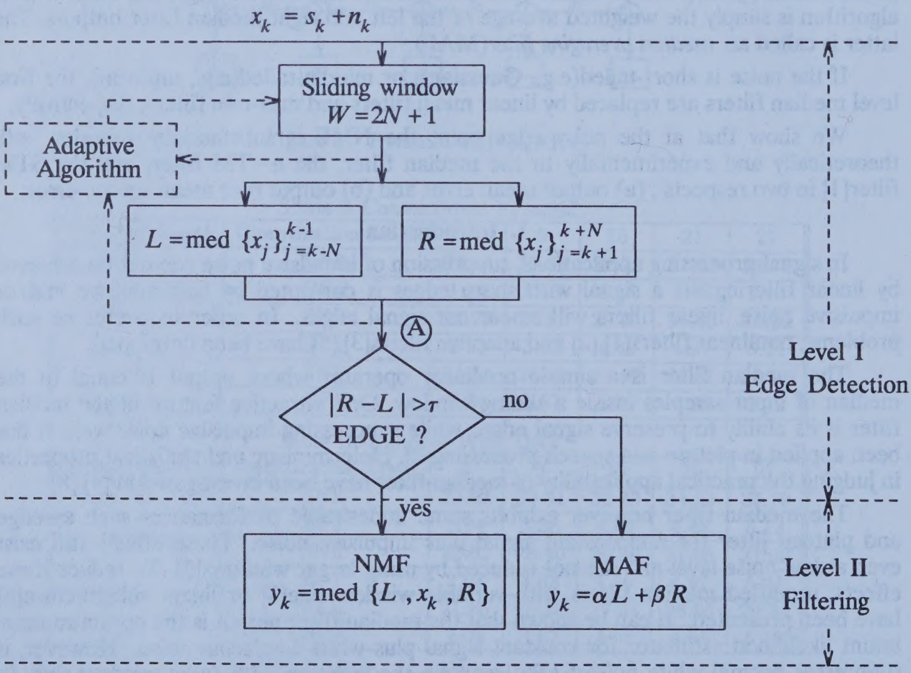
In signal processing applications, suppression of impulsive noise can not be achieved by linear filtering. If a signal with sharp edges is corrupted by nonimpulsive and/or impulsive noise, linear filters will smear out signal edges. In order to overcome such problems, nonlinear filters[1],[2] and adaptive filters[3],[5] have been developed.

The median filter is a simple nonlinear operator whose output is equal to the median of input samples inside a sliding window. One attractive feature of the median filter is its ability to preserve signal edges while suppressing impulsive noise well. It has been applied in picture and speech processing[2]. Deterministic and statistical properties in judging the practical applicability of median filter have been investigated in [4],[8].

The median filter however exhibits some undesirable performances such as edge and plateau jitter for nonconstant signal plus impulsive noise. These effects still exist even at low noise level and are not reduced by using larger windows[5]. To reduce these effects, modified median filters with variable window size[3] or linear substructure[6] have been presented. It can be shown that the median filter output is the optimum maximum likelihood estimator for constant signal plus white Laplacian noise. However, it suppresses normal white noise 57 % less than the averager with same window size. In general, it does not provide sufficient smoothing of nonimpulsive noise components. To overcome this, other filtering schemes have been proposed. One of them is to use a median filter in combination with linear filters[6]. This scheme generates fairly good performance in speech processing. However, the low-pass linear filters used as substructures always smear out signal edges. Bovik, Huang and Munson[7] introduced an order statistic filter whose output is a linear combination of ordered samples inside a window. That filter becomes a compromise between the median filter and the averager in noise suppression when the desired signal is constant. Motivated by robust estimation needs, Lee and Kassam[1] described another generation of median related filters.

These filters can suppress nonimpulsive noise quite well, but still have edge jittering problems. Signal edges corrupted by impulsive noise near edges or nonimpulsive noise can be blurred after the median filtering and median-related filtering. An one-dimensional edge enhancing comparison and selection filter has been introduced. It shows good performance in edge enhancing for deterministic signal but it seems to be vulnerable to noise[5].

In order to overcome edge shifting by median and median related filters, we need to detect the edge location and use this information for filtering. Recently, Lee and Tantaratana[5] introduced the *modified median filter*(MMF) based on detection of the edge location. If an edge is detected, the output of MMF is replaced by the median output based on a comparison of median value to the current sample value. They showed that the MMF can retain more fine details than can the median filter with same window size, as well as reduce the output mean square error which is caused mainly by edge shifting. To suppress nonimpulsive noise as well as impulsive noise, the *decision-based order statistic filter*(DBOSF) which is a combination of MMF with L-type structure of [1] was also presented in [5]. However, these filters are based on the assumption that there is at most one impulse inside the window. Therefore, under high-density impulsive noise, they might perform poorly.



$x_k = s_k + n_k$
Sliding window
$W = 2N+1$
Adaptive Algorithm
$L = \text{med}\ \{x_j\}_{j=k-N}^{k-1}$
$R = \text{med}\ \{x_j\}_{j=k+1}^{k+N}$
Ⓐ
$|R - L| > \tau$
EDGE ?
no
yes
Level I
Edge Detection
NMF
$y_k = \text{med}\ \{L, x_k, R\}$
MAF
$y_k = \alpha L + \beta R$
Level II
Filtering

Fig. 1 A block diagram for adaptive edge detection based composite nested median filter and median averaging filter

In this paper, we introduce a nested median filter and a median averaging filter as an adaptive edge detection based median-type filter. The proposed structure, as shown in Fig. 1, consists of two levels. The first level constitutes the edge detection stage. The windowed data is passed on to a pair of median filters looking to the left and right of the test point. This pair of median filters eliminate impulsive noise while providing estimates of the signal levels at either side of the point being tested for an edge. The outputs of the median filters are differenced and compared with a threshold in the decision block.

As shown in Fig. 1, if an edge detected, the left and right median filter outputs and the test point itself are inputted to another median filter. We call this second filter, a *nested median filter*(NMF), since it operates as a median of median. If the test point is not an edge, the left and right median filter outputs are averaged as indicated in the *median averaging filter*(MAF). A threshold τ is chosen by the maximum likelihood ratio test associated with Bayesian minimum error criterion.

If the noise is Gaussian, or uniform distributed, the first level median filters are replaced by linear mean filters and min-max filters, respectively. The NMF preserves the detected edge point without edge shifting while eliminating impulse, if present, at that point. In the absence of an edge, the MAF replaces the noisy test point by an average of the impulse-free outputs of the L and R median filters.

2. The Two – Level Filter Structure

The signal model assumes that the noise-free signal consists of two possible regions inside a sliding window used for filtering : (1) a homogeneous region – all the samples inside the window have the same value s, (2) an edge region – the sample inside window assumes one of two possible values, s_h and s_l with height $h = s_h - s_l > 0$. If the window size is small enough compared to that of the entire signal to be filtered, these assumptions are reasonably valid for practical purposes. The noise n_k is additive zero mean iid impulsive white having Laplacian or Cauchy or a mixture of Gaussian and Cauchy distribution. In this Section, we will describe procedures for edge detection and for NMF based on edge detection.

2.1 Edge Detection

Consider an step edge with height h inside window shown in Fig. 2

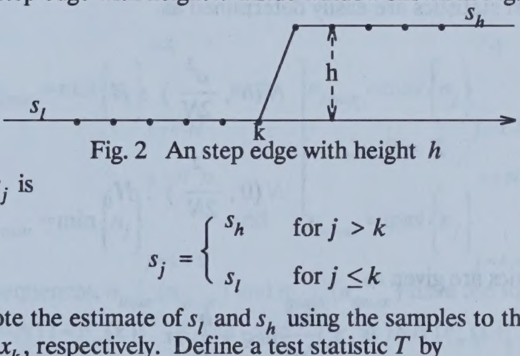


Fig. 2 An step edge with height h

where, the signal s_j is

$$s_j = \begin{cases} s_h & \text{for } j > k \\ s_l & \text{for } j \le k \end{cases}$$

Let L and R denote the estimate of s_l and s_h using the samples to the left and right side of the test sample x_k, respectively. Define a test statistic T by

$$T \triangleq R - L$$

With such an edge model we now make the following hypotheses

$$H_1 : | T | > \tau \quad x_k \text{ is on edge}$$

$$H_0 : | T | < \tau \quad x_k \text{ is not on edge}$$

The two considerations in the proposed edge detector are the left and right estimators and the threshold setting. The optimum maximum likelihood estimator(MLE) is the median, the average, and the average of maximum and minimum sample values, for Laplacian, Gaussian, and uniform noise pdf's, respectively. The threshold τ is also chosen according to the MLE measure. Under alternative hypothesis H_1,

$$x_j = \begin{cases} s_l + h + n_j & \text{for } j > k \\ s_l + n_j & \text{for } j \le k \end{cases}$$

Under null hypothesis H_0,

$$x_j = s_l + n_j \quad \text{for all } j$$

Without loss of generality, s_l can be assumed to be zero. Let $f(T \mid H_1)$ and $f(T \mid H_0)$ denote the pdf of T under H_1 and H_0, respectively. Then, the average probability of error can be expressed as

$$P_E = (1-P_D)Pr(H_1) + P_F Pr(H_0) \tag{2.1}$$

where P_F, P_D are respectively the probabilities of false alarm and of correct detection. The likelihood function $\rho(T)$ to minimize P_E can be obtained as

$$\rho(T) \triangleq \frac{f(T \mid H_1)}{f(T \mid H_0)} \gtrless \frac{Pr(H_0)}{Pr(H_1)} = q \tag{2.2}$$

The application of this statistical decision rule to determine the threshold value requires knowledge of the conditional pdf of T and a priori probabilities, $Pr(H_0)$ and $Pr(H_1)$. The latter can be estimated from the class of signals under analysis.

2.1.1 Mean Estimate Based Edge Detection

Let $\{n_j\}$ be a random sample sequence from an iid Gaussian distribution with zero mean and variance σ^2. Then, the sample mean estimators, L and R, are given by

$$L = \frac{1}{N}\sum_{i=1}^{N} x_{k-i} \quad \text{and} \quad R = \frac{1}{N}\sum_{i=1}^{N} x_{k+i} \tag{2.3}$$

For this case, the test statistics are easily determined as

$$T = R - L \sim \begin{cases} N(h, \dfrac{\sigma^2}{2N}) : H_1 \\ \\ N(0, \dfrac{\sigma^2}{2N}) : H_0 \end{cases}$$

The sufficient statistics are given as

$$T \gtrless \frac{2\sigma^2}{Nh}\ln q + \frac{h}{2} = \tau \tag{2.4}$$

2.1.2 Median Estimate Based Edge Detection

Let $\{n_j\}$ be a random sample sequence from iid white noise with a zero mean heavy-tailed distribution like Laplacian or Cauchy noise with distribution $F_n(x)$ and pdf $f_n(x)$. The median estimates follow as :

$$L = \operatorname{med}\left\{x_j\right\}_{j=k-N}^{k-1} \quad \text{and} \quad R = \operatorname{med}\left\{x_j\right\}_{j=k+1}^{k+N} \tag{2.5}$$

Under alternative hypothesis H_1,

$$L = \operatorname{med}\left\{n_j\right\}_{j=k-N}^{k-1} \quad \text{and} \quad R = h + \operatorname{med}\left\{n_j\right\}_{j=k+1}^{k+N}$$

The pdf of median output for iid N input samples is given by

$$f_L(x)=\frac{N!}{M!\,M!}F_n^M(x)[1-F_n(x)]^M f_n(x)\ ,\quad N=2M+1$$

Therefore, the conditional pdf's of T are

$$f(T \mid H_j)=\int_{-\infty}^{\infty} f_R(T+x)f_L(x)dx$$

where $f_R(x)$ is respectively $f_L(x)$, $f_L(x-h)$, under H_0, H_1. By substituting $f(T \mid H_1)$ and $f(T \mid H_0)$ into equation (2.2) the optimum threshold value can be obtained numerically for given q and $F_n(x)$. In particular, for $q=1$, the approximate optimum threshold value τ becomes $h/2$ when the noise under analysis is Laplacian or Cauchy.

2.1.3 Min – Max Estimate Based Edge Detection

Let $\{n_j\}$ be a random sample sequence from iid zero mean white noise with middle-tailed distribution, like uniformly distributed noise. Then

$$L=\frac{x_{lmin}+x_{lmax}}{2} \quad \text{and} \quad R=\frac{x_{rmin}+x_{rmax}}{2} \tag{2.6}$$

where $x_{lmin}(x_{lmax})$ and $x_{rmin}(x_{rmax})$ are the minimum(maximum) value of the left and right side samples from x_k. Under alternative hypothesis H_1,

$$L=\frac{n_{lmin}+n_{lmax}}{2} \quad \text{and} \quad R=h+\frac{n_{rmin}+n_{rmax}}{2}$$

where

$$n_{lmin}=\min\left\{n_j\right\}_{j=k-N}^{k-1} \quad \text{and} \quad n_{lmax}=\max\left\{n_j\right\}_{j=k-N}^{k-1}$$

and

$$n_{rmin}=\min\left\{n_j\right\}_{j=k+1}^{k+N} \quad \text{and} \quad n_{rmax}=\max\left\{n_j\right\}_{j=k+1}^{k+N}$$

Since $\{n_j\}$ are iid sequences, $n_{lmin}(n_{lmax})$ and $n_{rmin}(n_{rmax})$ have the following same pdfs.

$$f_{n_{\min}}(x)=N[1-F_n(x)]^{N-1}f_n(x) \quad \text{and} \quad f_{n_{\max}}(x)=N[F_n(x)]^{N-1}f_n(x)$$

By denoting n_m as the sum of n_{lmin} and n_{lmax}, the pdf of n_m becomes

$$f_{n_m}(x)=\int_{-\infty}^{\infty} f(x\text{-}y,y)dy \text{ with } f(x,y)=\begin{cases} N(N\text{-}2)[F_n(y)-F_n(x)]^{N\text{-}2}f_n(x)fn(y) & \text{for } x>y \\ 0 & \text{for } x \le y \end{cases}$$

Thus, the conditional pdf of T under H_1 becomes

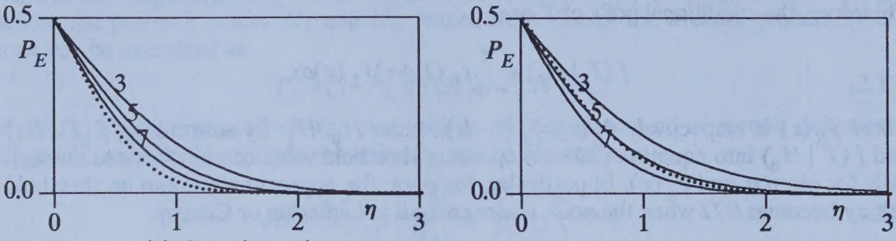
$$f(T \mid H_1)=\int_{-\infty}^{\infty} f_R(T+x)f_L(x)dx$$

where $f_L(x)=2f_{n_m}(2x)$ and $f_R(x)=f_L(x-h)$. Similarly, under null hypothesis H_0

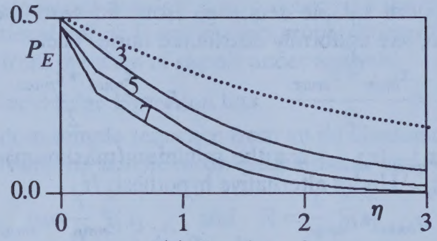
$$f(T \mid H_0)=\int_{-\infty}^{\infty} f_R(T+x)f_L(x)dx$$

where $f_R(x)=f_L(x)=2f_{n_m}(2x)$. If the noise is uniformly distributed between $-1/2$ and

1/2, then the approximate optimum threshold value τ becomes $h/2$ when $q=1$. The probability error in equation (2.1) based on estimators of signal edge levels under several noise distributions is shown in Fig. 3 . Clearly, the median based detector is optimal for Laplacian and Cauchy noise, while the mean based detector is the choice for Gaussian.



(a) Gaussian noise (b) Laplacian noise



(c) Cauchy noise

Fig. 3 Probability of error of edge detector ; solid curve indicates median filter with size $N=3,5,7$; dotted curve for mean filter with $N=7$

2.2 The Nested Median Filter and Median Averaging Filter

Based on the edge detector of Section 2.1, we propose the NMF strategy for edge point preservation and the MAF for non-edge point smoothing. To preserve details of signal and to prevent edge shifting due to impulsive noise near an edge, the median filter with small window size along an edge region is preferred. Therefore, the NMF at detected noisy edge is chosen to be the 3 point median

$$y_k = \text{med } \{L , x_k , R\}$$

If the noise is Gaussian-like so that the two estimates, L and R, are sample means, our y_k becomes the same as the output of FMH[6] which is used without an edge detector. However, if the noise is impulsive or uniformly distributed, then our filter is more effective than the FMH in preserving details and attenuating noise along edge region as shown in Section 4.

When no edge is detected, we use the weighted average of forward and backward estimates, L and R, for y_k. The output of MAF is

$$y_k = \alpha L + \beta R$$

where α and β are weighting coefficients and they are chosen optimally to minimize mean square error under unbiased estimator condition. The block diagram for the composite NMF and MAF is shown in Fig. 1 .

2.3 The Adaptive Algorithm

The success of our algorithm depends on the elimination of impulses in the L and R median filters. An adaptive algorithm for the detection of an impulse in L or R is shown in the substructure in Fig. 4 . We detect the presence of an impulse whenever $|L-\hat{s}_{k-M}|>\tau_I$ or $|R-\hat{s}_{k+M}|>\tau_I$ and then increase the window size N to $N+2$.

The estimates $\hat{s}_{k-M}$ and $\hat{s}_{k+M}$ are themselves outputs of median filters :

$$\hat{s}_{k-M} = \text{med}\left\{x_j\right\}_{j=k-M-\frac{Q_L}{2}}^{k-M+\frac{Q_L}{2}} \quad \text{and} \quad \hat{s}_{k+M} = \text{med}\left\{x_j\right\}_{j=k+M-\frac{Q_L}{2}}^{k+M+\frac{Q_L}{2}}$$

We set the windows Q_L and Q_R to force $\hat{s}_{k-M}$ and $\hat{s}_{k+M}$ to be free of impulses by the following logic. For simplicity, we will describe the logic for $\hat{s}_{k-M}$ only.

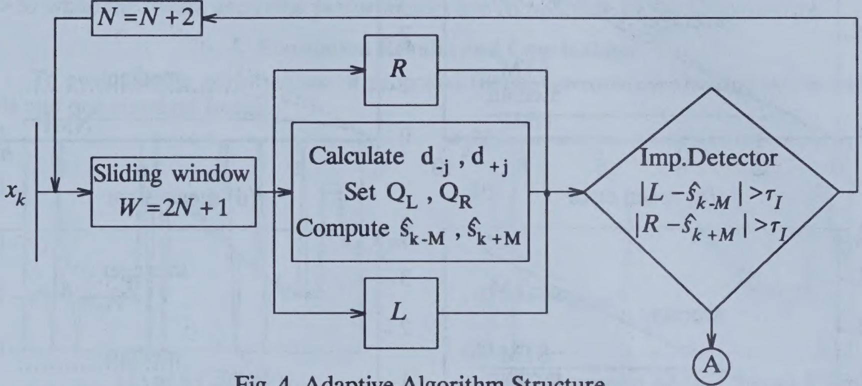

$N=N+2$
R
x_k
Sliding window
$W=2N+1$
Calculate d_{-j} , d_{+j}
Set Q_L , Q_R
Compute $\hat{s}_{k-M}$, $\hat{s}_{k+M}$
Imp.Detector
$|L-\hat{s}_{k-M}|>\tau_I$
$|R-\hat{s}_{k+M}|>\tau_I$
L
A

Fig. 4 Adaptive Algorithm Structure

Define the following test statistics for $j=1 \to 3$

$$d_{+j} \triangleq x_{k-M+j-1} - x_{k-M+j} \qquad d_{-1} \triangleq x_{k-M+1-j} - \hat{s}_{k-M-1}$$

Then,

if $(d_{+3}>\tau_3$ and $d_{-1}>\tau_3)$ or $(d_{+3}<-\tau_3$ and $d_{-1}<-\tau_3)$
choose $Q_L=7$
if $(d_{+2}>\tau_2$ and $d_{-1}>\tau_2)$ or $(d_{+2}<-\tau_2$ and $d_{-1}<-\tau_2)$
choose $Q_L=5$
if $(d_{+1}>\tau_1$ and $d_{-1}>\tau_1)$ or $(d_{+1}<-\tau_1$ and $d_{-1}<-\tau_1)$
choose $Q_L=3$
otherwise
choose $Q_L=1$

These equations are an extension of Lin's[3] adaptive algorithm to handle both positive and negative impulses simultaneously. If $Q_L=1$, rather than do no filtering, we instead use a sample mean with censored data using x_{k-M} as the reference point. This output gives $\hat{s}_{k-M}$ as the sample mean of $x_j \in L_c$, where

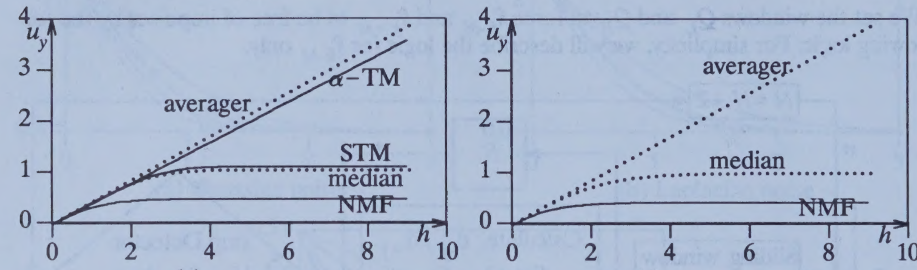
$$L_c = \{x_{k-M} - c \le x_j \le x_{k-M} + c \ , \ c = 3\sigma \ , \ j \in (k-M-3, k-M+3)\}$$

3. Some Statistical Properties of NMF

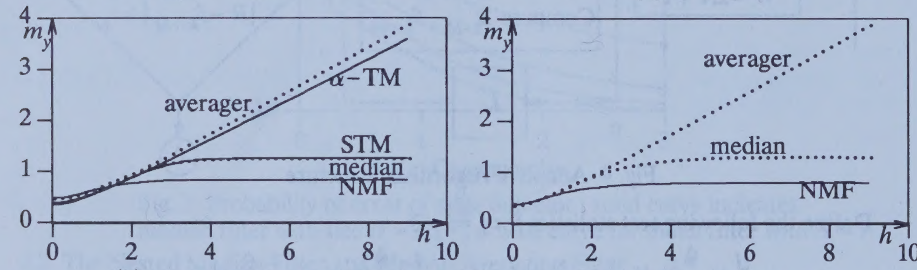
We discuss some statistical properties of the proposed NMF at noisy edge sample point. Consider an noise corrupted step edge with $s_l=0$ such as Fig. 2 . If the output y_k of NMF is taken over three values L , x_k , and R which are independent but not identically distributed, then we can get the pdf of y_k. The considered performance measures on filter output at the noisy edge are

$$\text{mean error}(u_y) = E\{y_k\}$$
$$\text{mean square error}(m_y^2) = E\{(y_k - s_k)^2\} = E\{y_k^2\}$$
$$\text{variance}(\sigma_y^2) = E\{y_k^2\} - u_y^2$$

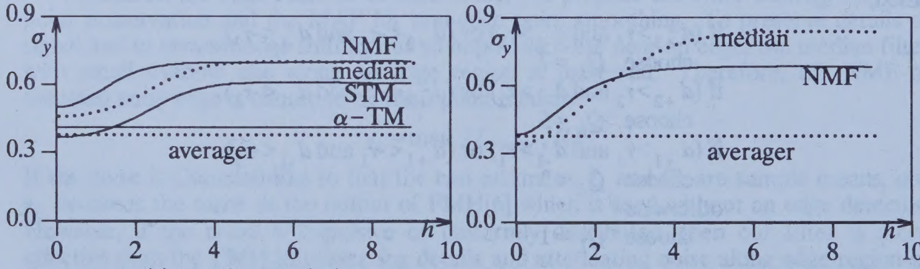
Using the pdf for y_k and the estimators in Section 2.2, we can calculate the above measures under different types of noise.



(a) mean error (d) mean error



(b) root mean square error (e) root mean square error



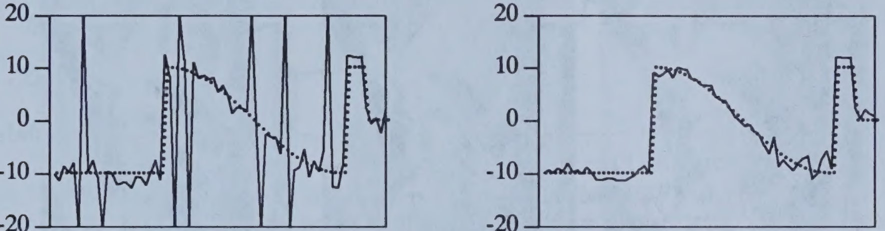
(c) standard deviation (f) standard deviation

Fig. 5 Output mean, rmse, and standard deviation for NMF with size 7 for edge of height h under ; (a)-(c) Gaussian noise of zero mean and unit var. ; and unit var. ; (d)-(f) Laplacian noise of zero mean and unit var.

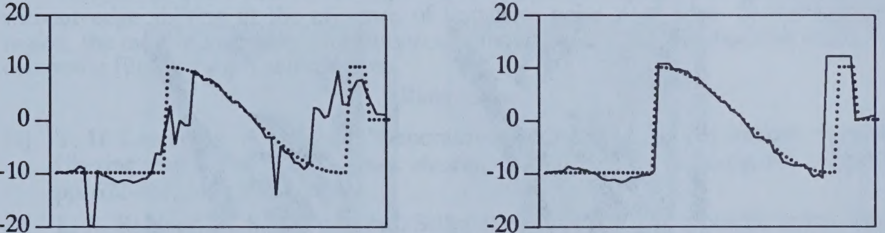
Fig. 5(a)-(c) show the output mean error, rmse, and standard deviation performance measures for NMF with window $2N+1=7$ versus input edge height h under Gaussian noise of zero mean and unit variance. These are compared with those of the averager, the median filter, the $\alpha-$TM filter and the STM filter with the same window size. Fig. 5(a) shows that the output mean error of NMF is bounded as it is for the median filter and the STM filter as h goes to ∞, and is less than or equal to those of other filters. This result indicates the improved edge preserving performance. For $h > \sigma$, the NMF can reject influence of the samples from the other level of edge as outliers better than the median filter and STM filter. This outlier resistance allows the NMF to limit the influence of the samples from across the edge. The results shown in Fig. 5(b) confirm that minimizing mse emphasizes noise suppression for small edges, favoring the mean filter, and edge preserving for large edges, favoring the NMF. Under Gaussian-like noise and small edge, the edge detector leads to the MAF processor whose performance is similar to the standard averager. Therefore, the composite NMF exhibits robust output mse performance over small to large edges. Fig. 5(c) shows that the variance of NMF is greater than those of the other filters but becomes substantially same as that of the median filter. This is expected since the averager gives the best performance in Gaussian noise. The corresponding performance measures for the impulsive noise case are shown in Fig. 5(d)-(f). Here, the variance of the NMF is less than that of the median filter for $h > 3\sigma$ while the mean, and rmse performances are comparable to the Gaussian noise.

4. Simulation Results and Conclusions

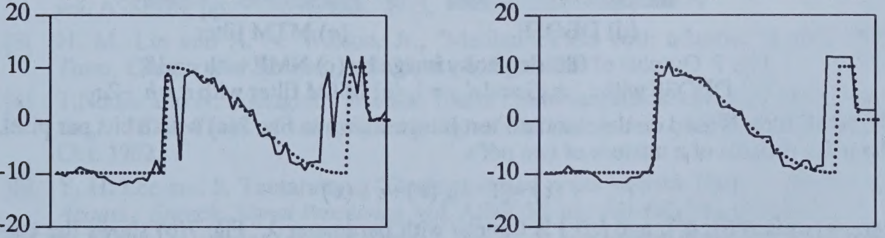
To evaluate the performance of proposed filters, we test two one-dimensional signals and one standard image.



(a) noisy signal (b) composite NMF(rmse=1.3729)



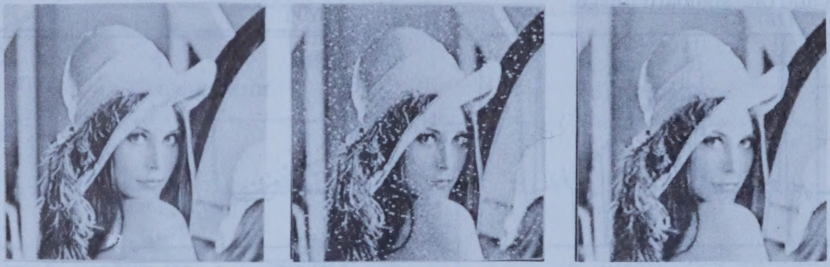
(c) α−TM filter(rmse=4.5357) (d) MTM filter(rmse=3.6478)



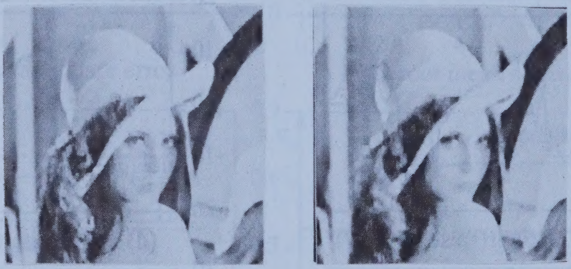
(e) FMH filter(rmse=3.7985) (f) median filter(rmse=3.5980)

Fig. 6 Outputs of filtering input by (b) NMF with $\tau=5$, (c) $\alpha-$TM filter with $\alpha=0.2$, (d) MTM filter with $q=h-2\sigma$, (e) FMH filter, and (f) median filter

The first test input[1] shown as solid curve in Fig. 6(a) was obtained by Gaussian noise of standard deviation $\sigma=2$ and random impulse of height ±30 with probability 0.15 to an original signal of 70 samples shown as the dotted curve in Fig. 6(a). The original signal has a constant part, a sinusoidal part, and a narrow pulse of duration 4 samples with the minimum edge height $h=10$. Fig. 6(c) -(f) shows the result of filtering the test input by the α-TM filter with $\alpha=0.2$, the MTM filter with $q=h-2$, the FMH filter with 3-point averager, and the median filter, respectively. The window size was 7 because the narrow pulse with duration 4 samples will be preserved by the median filter of window size 7. It can be seen that the α-TM filter and the FMH filter smear edges severely and fail to suppress impulsive noise sufficiently while the MTM filter and the median filter suppress impulsive noise quite well. However, since there exist impulsive noise near edges the median output results in the edge shifting at the two parts of edge, which might be expected. Even though the MTM filter can suppress noise along a constant part and a sinusoidal part better than the median filter, it can not reduce such edge shifting problems. Fig. 6(b) is the result of filtering the same input by the composite NMF of same window size with threshold $\tau=5$. Since this filter is based on edge detection, no edge shifting occurs at all for this test input. It can suppress noise reasonably well but not as much as the MTM filter does. The proposed filter gives much smaller sample rmse than others because rmse is caused mainly by edge shifting.



(a) original image (b) noisy image (c) composite NMF



(d) DBOSF (e) MTM filter

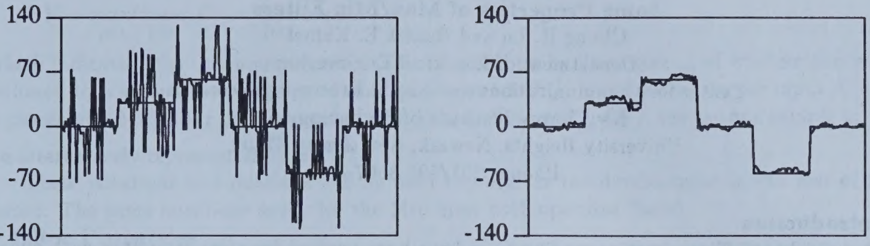
Fig. 7 Outputs of filtering noisy image by (c) NMF with $\tau=15$, (d) DBOSF with $t_{th}=15$ and $d_{th}=1$, (e) MTM filter with $q=h-2\sigma$

The NMF filter is used on the standard test image shown in Fig. 7(a) with 8 bits per pixel. The noise consists of a mixture of two pdf's.

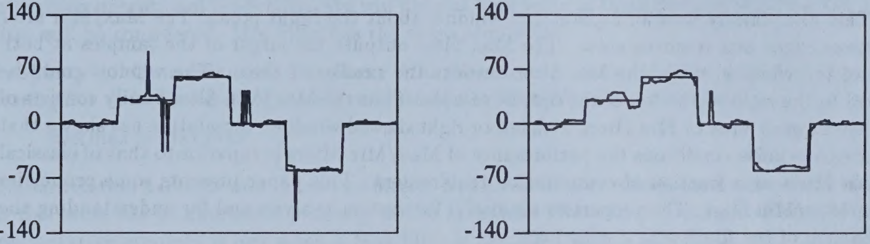
$$f(x)=(1-\varepsilon)g(x)+\varepsilon h(x)$$

Here, $g(x)$ is $N(0,\sigma^2)$, and $h(x)$ is Cauchy with parameter λ. Fig. 7(b) shows the corrupted image with $\varepsilon=0.4$, $\sigma=10$, and $\lambda=8$. To estimate signal levels at either side of an edge, we use three pixels closest to the current sample for the left or right median filters. Fig. 7(c) shows the NMF filtered image(with $\tau=15$). For comparison, the DBOSF and the MTM filter for the same input image are shown in Fig. 7(d)-(e). It can be seen that the NMF preserves edges and suppresses impulsive noise better than the others.

Fig. 8 compares the adaptive NMF with the DBOSF and the median filter. Fig. 8(b) shows output of the adaptive composite NMF with $\tau=15$ and $\tau_I=15$. These results demonstrate the superiority of the adaptive NMF under high density of impulsive noise.



(a) noisy signal (b) adaptive NMF



(c) DBOSF (d) median filter

Fig. 8 Outputs of filtering input by (b) Adaptive NMF with $\tau = 15$, (c) DBOSF filter with $t_{th} = 15$, $d_{th} = 1$, and(d) median filter

We conclude that the nested median filter has the capability of preserving edge without edge shifting in the presence of impulsive noise near edge. In homogeneous region, the median averaging filter suppresses impulsive and/or nonimpulsive noise. The composite filter is robust and adaptive.

References

[1] Y. H. Lee and S. A. Kassam, "Generalized median filtering and related nonlinear filtering techniques," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-33, pp. 672-683, June 1985.

[2] L. R. Rabiner, M. Sambur, and C. Schmidt, "Applications of a nonlinear smoothing algorithm to speech processing," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-23, pp. 552-557, Dec. 1975.

[3] H. M. Lin and A. N. Willson, Jr., "Median Filters with adaptive length," *IEEE Trans. Circuits and Systems,* vol. 35, no. 6, June 1988

[4] T.Nodes and N. Gallagher, "Median filters : Some modifications and their properties," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-30, pp. 739-746, Oct. 1982.

[5] Y. H. Lee and S. Tantaratana, "Decision-based order statistic filters," *IEEE Trans. Acoust., Speech, Signal Processing* vol. ASSP-38, pp. 406-420, March 1990.

[6] P. Heinonen and Y. Neuvo, "FIR-median hybrid filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-35, pp. 832-838, June 1987.

[7] A. C. Bovik, T. S. Huang , and D. C. Munson, Jr., "A generation of median filtering using linear combination of order statistics," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-31, pp. 1342-1349, Dec. 1983.

[8] S. R. Peterson, Y. H. Lee, and S. A. Kassam, "Some Statistical Properties of Alpha-Trimmed Mean and Standard Type *M* Filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 707- 713, May 1988.

Some Properties of Max/Min Filters

Chung H. Lu and Raafat E. Kamel
Department of Electrical Engineering
Center for Communications and Signal Processing Research
New Jersey Institute of Technology
University Heights, Newark, New Jersey 07102
Phone: 201/596-3507

1. Introduction

Rank-order based filters, such as median filter have been applied for noise reduction with edge preserving capability. A new rank-order based Max/Min filter family was recently introduced [1]. This filter family uses an asymmetric window about the input point. The Max/Min filter preserves edges and removes noise. The Max filter outputs the larger of the samples at both ends of the window, while the Min filter outputs the smaller of them. The window could be skewed to the right or the left of the output sample. Thus the Max/Min filter family consists of four filter types: Max or Min filters with left or right skewed window. Simulation has shown that under severe noise conditions the performance of Max/Min filters is superior to that of classical median filters at a fraction of computation requirement. This paper presents some properties of the Max/Min filter. The properties are useful for system analysis and for understanding the mechanism of the filters.

Median filter and other rank-order based filters were studied extensively in literature[2-5]. These use symmetric windows. As the window size increases more computations are needed to calculate the output. The Max/Min filter family is a new type of rank-order based nonlinear filter, which requires one comparison per output irrespective of the window size. It has good filtering performance under severe noise conditions, including edge preserving property.

Besides noise reduction and edge preservation, Max/Min filter has an interesting effect. It shifts either rising or falling edges while maintaining the edge shape. The amount an edge is shifted is equal to the window size less one sample. Cascading Max/Min filters will achieve more noise reduction, with each pass incurring additional edge shifting. This will wipe out the pulses after a finite number of passes. Such edge shifting can be restored by using alternate passes of Max and Min filters [1,6].

In this paper, we introduce several properties of the Max/Min filter family. These properties can be exploited to simplify implementation of Max/Min filters. Section 2 introduces some definitions and notations used in this paper. Four properties are introduced in Section 3. The proof of these properties are given in [6].

2. Definition and Notation

Denote a data sequence or time series by X. Specifically, let

$$X \triangleq \{x_i\} \triangleq \{\cdots, x_{i-1}, x_i, x_{i+1}, \cdots\}$$

Consider a Max filter with window size $n+1$. Denote such filter by the operator

$$\max_{(n)} \text{ or } \max_{(n)}$$

the subscript (n) indicates the window size less one. Let Y be the output of the filter to an input X. Then we have the following relation

$$Y = \max_{(n)}(X) \text{ or } \{y_i\} = \max_{(n)}(\{x_i\})$$

where the output samples y_i's are related to the input samples x_i's by

$$y_i \triangleq \max(x_i, x_{i+n})$$

The filters may be cascaded such as

$$Y = \underset{(m)}{\max}(\underset{(n)}{\max}(X)) = \underset{(m)}{\max}\,\underset{(n)}{\max}(X)$$

which indicates that the input sequence X is operated on by filter $\max_{(n)}$ of window size $n+1$ followed by a second filter $\max_{(m)}$ of window size $m+1$. If Y is the output to the input X with p passes of a Max filter of window size $m+1$ then, $Y = \max^{p}_{(m)}(X)$, the output sample y_i can be alternatively represented by $x_i^{(m)(p)}$.

These notations and relations will be used throughout the development in the rest of this paper. The same notations apply for the Min filter with operator "min".

3 Filter Properties

Besides the edge preserving properties described previously, the Max/Min filters possess some interesting operational properties. We will present here some of these properties. Only the Max filter will be considered. Min filter has the same properties.

Property 1: Commutative Property *Max filters can be cascaded. The output of the composite filter is independent of the order in which the Max filters are cascaded.*

$$\underset{(m)}{\max}\,\underset{(n)}{\max} = \underset{(n)}{\max}\,\underset{(m)}{\max}$$

Similar relation holds for more than two filters. Nonlinear filters are in general not commutative. In this sense the commutative property of the Max/Min filters is very unique. Note that this commutative property is lost when a Max filter is cascaded with a Min filter.

Property 2: *A p-pass Max filter of window size $m+1$ is equivalent to a $(p+1)$-ary maximum filter of window size $pm+1$.*

$$x_i^{(m)(p)} = \max(x_i, x_{i+m}, \cdots, x_{i+pm})$$

Property 3: *The cascade of k Max filters whose window sizes are consecutive multiples of a base number n, is equivalent to a p-pass Max filter of window size n.*

$$\underset{(n)}{\max}\,\underset{(2n)}{\max}\cdots\underset{(kn)}{\max} = \underset{(n)}{\overset{p}{\max}}$$

where $p = 1+2+3+\cdots+k$.

Property 4: Decomposition Property *A p-pass Max filter of window size $m+1$ is equivalent to a cascade of Max filters with window sizes $b_j m+1$, $j = 1, 2, ..., k$.*

$$\underset{(m)}{\overset{p}{\max}} = \underset{(b_1 m)}{\max}\,\underset{(b_2 m)}{\max}\cdots\underset{(b_k m)}{\max}$$

where the numbers $b_1, b_2, ..., b_k$ are determined with the following rules:

(i) $b_1 = \lceil p \div 2 \rceil$,

(ii) $b_j = \lceil (p - \sum_{i=1}^{j-1} b_i) \div 2 \rceil$ *for* $j > 1$ *and satisfy the relation* $\sum_{i=1}^{k} b_i = p$ *where* $\lceil x \rceil$ *is the integer greater or equal to* x.

References

[1] C. H. Lu and R. E. Kamel, "Max/Min: A Novel Rank-Order Based Filter Family," *Proc. 1990 DSP Workshop*, New York, Sept. 1990.

[2] N. C. Gallagher, "Median Filter: A Tutorial," *1988 IEEE Inter. Symp. on Circuits & Systems*, Vol.2, pp. 1737-1744.

[3] S. J. Ko, *et al*, "Selective Median Filters," *1988 IEEE Inter. Symp. on Circuits & Systems*, Vol. 2, pp. 1495-1498.

[4] N. C. Gallagher, *et al*, "A Theoretical Analysis of The Properties of Median Filters," *IEEE Trans. Acoust., Speech & Signal Processing*, Vol. ASSP-29, pp. 1136-1141, Dec. 1981.

[5] G. R. Arce, and M. P. Mc Loughlin, "Theoretical Analysis of The Max/Median Filter," *IEEE Trans. Acoust., Speech & Signal Processing*, Vol. ASSP-35, No.1, pp. 60-69, Jan. 1987.

[6] R. E. Kamel, *Max/Min: A Novel Rank-Order Based Filter Family*, M.S. Thesis, New Jersey Institute of Technology, Newark, New Jersey Sept. 1990.

On the Conservation Law and the Performance Space of Single Server Systems

L. Georgiadis
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
I. Viniotis
North Carolina State University, Raleigh, NC 27695

Abstract

The conservation law and the achievable performance region for average response times of single server systems have been known for a long time. Currently, the description of the conservation law is restricted to the class of ergodic policies or policies with a regenerative structure, which guarantees steady state. However, this class does not include important policies, such as adaptive policies, which may base their decisions on the whole history of the system. For this class of policies, convenient regeneration points do not exist and therefore, ergodicity and existence of steady state is not guaranteed or at least is not easy to prove. In this paper we prove a conservation law for the Cesaro sums of response times without any ergodicity assumptions. We thus extend previous results by removing some unnecessary restrictions on the class of admissible policies. We also provide a characterization of the space of achievable response times for policies in this larger class.

1 Introduction.

Conservation laws regarding average response times and number of customers in the system have been known for a long time. Most of the conservation laws presented so far [8], deal with single server, *work conserving* systems, or multiple server, *strongly work conserving* systems [8]. Roughly speaking, in such systems, no work (service time) is created or destroyed within the system (e.g., customers do not balk, the server maintains always the same speed, etc.) For such systems, under ergodicity assumptions, one can prove invariance relations of the form (see e.g., [8],)

$$\sum_{i=1}^{N} \rho_i R_i(f) = F \qquad (1)$$

regarding, for example, the steady state statistical average of response times, under an arbitrary, *ergodic* priority discipline f. A number of inequalities [3] further constrain the set of achievable response times (i.e., the performance region) to be a convex polygon. In eq. (1), $R_i(f)$ is the response time for class i, ρ_i is the utilization of class i and F is a constant, independent of the scheduling policy. Relatively little can be said about multiple server systems [2].

Existence of steady state statistical averages (i.e., ergodicity under the priority rule used,) is a key assumption for all the above studies. The proof of ergodicity for a policy typically utilizes renewal arguments and is based on the existence of proper regeneration points (usually at the start of a busy period, the system regenerates [3].) However, for a large (and rather important for applications,) class of policies, namely adaptive [5], and in general, time and/or history dependent policies, such convenient regeneration points may not exist. Such algorithms may base their decisions on the entire history of the system, and therefore, at the beginning of a busy period (or, for that matter, at any time instant,) the system may not start statistically afresh.

Thus, it is worthwhile to investigate whether invariance relations hold for priority systems, *without* the above mentioned ergodicity assumptions. Since it may not be possible to show in advance that steady state distributions exist, the conserved quantities are now time (rather than statistical) averages. In this paper, we prove a conservation law for the Cesaro sums of response times under policies that are not necessarily ergodic.

The paper is organized as follows: in section 2 we define the $GI|G|1$ model for which the conservation law holds: we also specify the class of admissible policies and introduce some notation. In section 3 we first provide a counterexample to the claim that the Cesaro sums have a limit when the arrivals are Poisson. We then prove a conservation law for the Cesaro sums of response times induced by work-conserving policies. In section 4 we derive inequalities for the Cesaro sums of response times for the M/G/1 queue, thus characterizing further the achievable performance space. This extends the well known inequalities satisfied by the long run average response times induced by ergodic work-conserving policies. All missing proofs may be found in [6].

A few words on the notation that will be used in this paper. By $\{\Omega, \mathcal{F}, P\}$, we denote the probability space on which the stochastic system under consideration is defined. If X, Y are random variables or $\sigma-$fields, we denote by $\sigma(X,Y)$, the $\sigma-$field generated by X, Y. The indicator function of an event $A \in \mathcal{F}$ is denoted by I_A. The complement of a set R is denoted by R^c. If $f(\cdot)$ is a function with domain Δ, the range of values of $f(t)$ when $t \in R \subset \Delta$, is denoted by $\Re\{f(t); t \in R\}$. Also, we define $f^+(x) = \max\{0, f(x)\}$. Finally, when $i > j$, the expressions $\sum_i^j$ and $\cup_i^j$ are defined to be 0 and $\emptyset$ respectively.

2 System model and admissible policies.

We shall use the model described in [10]. More specifically, we consider a single server queue with N classes of customers. The time interval between the k^{th} and $k+1^{th}$ arriving customer ($k \geq 1$) is T_k. We assume that the time of the first arrival is zero. With each arrival there is an associated random variable, C_k, which denotes the class to which the arrival belongs; that is, if $C_k = i$, then the k^{th} customer belongs to class i. If the k^{th} arriving customer belongs to class i, its service time is a random variable S_{ik}. Therefore, the service time of the k^{th} arriving customer is given by

$$S_k = \sum_{i=1}^{N} I_{\{C_k=i\}} S_{ik}. \tag{2}$$

We assume that $\{T_k,\ k \geq 1\}$, $\{C_k,\ k \geq 1\}$, $\{S_{ik},\ k \geq 1\}$, $i = 1, \ldots, N$, are independent sequences. Each sequence contains i.i.d. random variables. It follows from (2) that

$\{S_k,\ k \geq 1\}$ is a sequence of i.i.d. random variables independent of $\{T_k,\ k \geq 1\}$. To avoid unnecessary complications we assume that $P(T_1 = 0) = 0,\ P(S_1 = 0) = 0$ and $P(C_1 = i) > 0,\ \ i = 1, \cdots, N$.

Note that if the random variables $\{T_k,\ k \geq 1\}$ are exponentially distributed, the described model is equivalent to the assumption that the arrival instants of each class constitute a Poisson process with rate $\lambda_i = P(C_1 = i)/ET_1$, independent of the arrival processes of the other classes. With these assumptions on the arrival processes, it is stated in [8, p. 432] that under *any* work-conserving policy (i.e., a nonidling policy that does not distinguish customers by service-time characteristics and does not affect customer interarrival times or service requirements), the long run averages of the response times exist. It was hinted in [2], however, that the previous statement may not be true. In section 3 we provide a counterexample which shows that indeed, it is possible to construct policies for which the long run averages of response times do not exist, even though the interarrival times of each class constitute independent Poisson processes.

In the sequel, we shall use the following quantities:

$p_i \triangleq P(C = i)$: probability an arriving customer belongs to class i.

$\lambda \triangleq 1/ET_1$:customer arrival rate (for all classes.)

$\lambda_i \triangleq p_i\lambda$: arrival rate of customers from class i.

$s_i^{(m)} \triangleq ES_i^m,\ s_i \triangleq s_i^{(1)}$

$\rho_i \triangleq \lambda_i s_i$: class i utilization.

S_{il} : service time of the l^{th} *arriving* customer from class i.

W_{il} : time in queue of the l^{th} *departing* customer from class i.

L_{il}: time instant when the service of the l^{th} *departing* customer from class i starts.

$\bar{S}_{il}$: service time of the l^{th} *departing* customer from class i.

L_l: time instant when the service of the l^{th} *departing* customer starts.

$\bar{S}_l$: service time of the l^{th} *departing* customer.

B_k : length of the k^{th} busy period.

Γ_k : time instant when the k^{th} busy period begins.

K_{im} : number of customers from class i served during the m^{th} busy period.

$M_{ik} \triangleq \sum_{m=1}^{k} K_{im}$

$V^{(\pi)}(t)$: work in system (i.e., sum of remaining service times of customers in the system at time t) when scheduling policy π is followed.

$V_i^{(\pi)}(t)$: work in system due to customers from class i at time t, when scheduling policy π is followed.

Nonpreemptive work-conserving scheduling policies do not depend on the knowledge of the service requirements of customers that are present in the system at the time when a scheduling decision is made. As a consequence, the service requirements of the successive *departing* customers from a given class (say class i), $\{\bar{S}_{ik}, k \geq 1\}$, are i.i.d random variables. In addition, $\bar{S}_{ik}$ is independent of the 'history' of the system up to time L_{ik}, and in particular independent of $W_{im}, 1 \leq m \leq k$. This claim is expressed formally in Lemma 1 below.

Let $\mathcal{A}(t)$ be the number of arrivals up to time t, i.e.,

$$\mathcal{A}(t) = \max\{k : \sum_{i=1}^{k} T_i \leq t\} + 1 \tag{3}$$

(Recall that a customer is assumed to arrive at $t = 0$.) Let also $\mathcal{C}(t)$ be the class index of the last customer to arrive up to time t, that is, $\mathcal{C}(t) = C_{\mathcal{A}(t)}$.

A *nonpreemptive* scheduling policy is characterized by a sequence of indices $\{J_k, k \geq 1\}$, where J_k denotes the *arrival* sequence number of the k^{th} served customer. J_k is related to L_k as follows:

1. $J_1 = 1$, $L_k = 0$,
2. $L_{k+1} \geq L_k + S_{J_k}$, $k \geq 1$, that is, the service of the $k+1^{th}$ customer may begin only if the service of the k^{th} customer ends.
3. $J_{k+1} \in \Re\{\mathcal{A}(t); t \leq L_{k+1}\} - \Re\{J_m;\ m = 1,2,\ldots,k\}$, $k \geq 1$, that is, the policy always serves one of the customers currently in the system.

We now define the class of nonanticipative nonpreemptive policies. Let $\wp(t)$, $t \geq 0\ldots$ be an increasing family of σ-fields independent of the sequences $\{T_k,\ k \geq 1\}$, $\{C_k,\ k \geq 1\}$, $\{S_{ik},\ k \geq 1\}$ ($\wp(t)$ is introduced here to allow for the possibility that a policy may be randomized), and define

$$\begin{aligned}
\mathcal{F}_0(t) &= \mathcal{G}(t) = \sigma(\mathcal{A}(\tau), \mathcal{C}(\tau); \tau \leq t, \wp(t)) \\
\mathcal{F}_k(t) &= \sigma(S_{J_m}; 1 \leq m \leq k, \mathcal{G}(t)).
\end{aligned}$$

A *nonanticipative* nonpreemptive scheduling policy has the following properties

1. L_k is a stopping time for $\mathcal{F}_{k-1}(t)$ for all $k \geq 1$, (that is, $\{L_k \leq t\} \in \mathcal{F}_{k-1}(t)$ for all $t \geq 0$.)
2. $J_k, k \geq 1$ is $\mathcal{F}_k$ measurable, where $\mathcal{F}_k$ is the σ-field of sets 'prior' to L_k, that is,

$$\mathcal{F}_k = \{A \in \mathcal{F} : A \cap \{L_k \leq t\} \in \mathcal{F}_{k-1}(t), \forall t \geq 0\}.$$

Using standard techniques (see, for example, [1, section 4.2]) it can be shown that $\mathcal{F}_k \subset \mathcal{F}_{k+1}$ and that L_k, C_{J_k} are $\mathcal{F}_k$-measurable , for all $k \geq 1$.

Let Q_{ik} be the *departing* sequence number of the k^{th} served customer from class i. For a nonanticipative nonidling policy, Q_{ik} is a stopping time for $\mathcal{F}_k$. This follows by induction after observing that

$$\begin{aligned}
\{Q_{i1} = n\} &= \{C_{J_n} = i\} \cap_{m=1}^{n-1} \{C_{J_m} \neq i\} \in \mathcal{F}_n, \ \forall n, \text{ and,} \\
\{Q_{ik} = n\} &= \cup_{m=k-1}^{n-1} [\{Q_{i(k-1)} = m\} \cap \{C_{J_n} = i\} \cap_{l=m+1}^{n-1} \{C_{J_l} \neq i\}].
\end{aligned}$$

Let $\mathcal{F}_{ik}$ be the σ-field of the sets prior to Q_{ik} (with respect to $\mathcal{F}_k$), that is,

$$\begin{aligned} \mathcal{F}_{ik} &= \{A \in \mathcal{F} : A \cap \{Q_{ik} = m\} \in \mathcal{F}_m, \ \forall m\} \\ &= \{A \in \mathcal{F} : A \cap \{Q_{ik} = m\} \cap \{L_m \le t\} \in \mathcal{F}_{m-1}(t), \ \forall t \ge 0, \ \forall m\} \end{aligned}$$

The following properties can be derived from the definitions.

Lemma 1 *Suppose that L_k and Q_{ik} are finite a.e.. Under any nonanticipative nonpreemptive policy,*

1. *W_{ik} is $\mathcal{F}_{ik}$-measurable for every $k \ge 1$.*
2. *$\bar{S}_{i(k-1)}$ is $\mathcal{F}_{ik}$-measurable for every $k \ge 2$.*
3. *$\bar{S}_{ik}$ is independent of $\mathcal{F}_{ik}$ for every $k \ge 1$.*
4. *For a fixed i, $1 \le i \le N$, the sequences $\{\bar{S}_{ik}, \ k \ge 1\}$ and $\{S_{ik}, \ k \ge 1\}$, are identically distributed.*

A *nonidling* policy never leaves the server idle when there is a customer in the system. A nonpreemptive policy is called *work-conserving* if it is nonidling and nonanticipative.

We are now in a position to state and prove the conservation law.

3 The conservation law.

In this section we present first a counterexample that shows that even when the arrival process is Poisson, ergodicity of the scheduling policies (i.e., existence of the long run averages) is not guaranteed. Stronger conditions on the scheduling policies must be imposed.

Counterexample. Consider an M/G/1 queue with two classes of customers. Let R^h (resp. R^l) be the long run average of response time of class 1 under the policy which always gives strict nonpreemptive priority to class 1 (resp. class 2). Assume that the system is stable (i.e., $\lambda_1 ES_1 + \lambda_2 ES_2 < 1$), the second moments of the service requirements are finite, and that $\lambda_1 ES_1 > 0$, $\lambda_2 ES_2 > 0$. Consider a time dependent policy π, that operates as follows:

> At the beginning of the n^{th} busy period ($n \ge 1$,) give highest priority to class 1 if $2^{2k} \le n \le 2^{2k+1} - 1$, for some $k \ge 0$. Otherwise, (i.e., if $2^{2k+1} \le n \le 2^{2(k+1)} - 1$), give highest priority to class 2.

Consider an $M|G|1$ system under policy π. Let $\mathcal{R}_n$, $n \ge 1$, be the sum of the response times of class 1 customers during the n^{th} busy period only. Define the sets of busy periods over which class 1 has highest and lowest priority as

$$H(n) = \{l : \text{ class 1 has highest priority during the } l^{th} \text{ busy period}, 1 \le l \le n\}$$

$$L(n) = \{l : \text{ class 1 has lowest priority during the } l^{th} \text{ busy period}, 1 \le l \le n\}$$

respectively. Of course, $H(n) \cup L(n) = \{1, 2, \cdots, n\}$. Let $h(n)$ (resp. $l(n)$) denote the cardinality of the set $H(n)$ (resp. $L(n)$). Then, R_n, the (time) average response time of customers from class 1, during the first n busy periods, is given by

$$R_n = \frac{1}{n}\sum_{i=1}^{n} \mathcal{R}_i = \frac{h(n)}{n}\frac{\sum_{i \in H(n)} \mathcal{R}_i}{h(n)} + \frac{l(n)}{n}\frac{\sum_{i \in L(n)} \mathcal{R}_i}{l(n)}.$$

Consider now the subsequence $\{n_k = (2^{2k} - 1),\ k \geq 1\}$. Observe that $h(n_k) = (2^{2k} - 1)/3$ and $l(n_k) = (2^{2k+1} - 2)/3$. Since the interarrival times are exponential, the random variables $\mathcal{R}_k,\ k \in H(n)$ (resp. $\mathcal{R}_k,\ k \in L(n)$) are i.i.d., identically distributed to the corresponding random variables in a queueing system that gives always nonpreemptive priority to customers in class 1 (resp. in class 2). Since $\lim_{k \to \infty} h(n_k) = \infty$ and $\lim_{k \to \infty} l(n_k) = \infty$, from the strong law of large numbers we have that $\lim_{k \to \infty} R_{n_k} = \frac{1}{3}R^h + \frac{2}{3}R^l$ *a.e.* Similarly, if we consider the subsequence $\{m_k = 2^{2k+1} - 1,\ k \geq 0\}$, we find that $h(m_k) = (2^{2(k+1)} - 1)/3$ and $l(m_k) = (2^{2k+1} - 2)/3$. Therefore, $\lim_{k \to \infty} R_{m_k} = \frac{2}{3}R^h + \frac{1}{3}R^l$ *a.e.* Since $R^h < R^l$, we see that the long run averages converge to two different points along the two subsequences $\{n_k\}, \{m_k\}$. Consequently, the long run average limits do not exist. □

The key to the arguments that follow is the following lemma, which can be derived from a Strong Law of Large Numbers for martingales [7]: [1]

Lemma 2 *Let $\{X_k,\ k \geq 1\}$, $\{Y_k,\ k \geq 1\}$ be sequences of random variables and $\{\mathcal{F}_k, k \geq 1\}$ be an increasing family of σ-fields. Suppose that X_k is $\mathcal{F}_k$-measurable for all $k \geq 1$ and that Y_{k-1} is $\mathcal{F}_k$-measurable for all $k \geq 2$. Suppose also that Y_k is independent of $\mathcal{F}_{k-1}$, for all $k \geq 1$. Let there be a random variable Z, a constant $\delta > 1$, and a constant c such that $E|Z|^\delta < \infty$ and $P(|X_k Y_k| > x) \leq cP(|Z| > x)$ for each $x \geq 0$ and $k \geq 1$. Then*

$$\lim_{k \to \infty} \left(\frac{\sum_{m=1}^{k} X_m Y_m}{k} - \frac{\sum_{m=1}^{k} X_m E Y_m}{k} \right) = 0 \quad a.e. \tag{4}$$

The next lemma, in conjunction with Lemma 1, shows that the sequences $X_k = W_{ik}, Y_k = \bar{S}_{ik},\ k \geq 1$ satisfy the conditions of Lemma 2.

Lemma 3 *Suppose that $\sum_{i=1}^{N} \rho_i < 1$ and that for some constant $\gamma > 2$, $s_i^{(\gamma)} < \infty$, $i = 1, \cdots, N$. Then under any work-conserving policy, there exists a constant c_i and a random variable $\bar{Z}_i$ such that $E\bar{Z}_i^{\gamma/2} < \infty$, and for each $x \geq 0$ and $k \geq 1$, we have that*

$$P(W_{ik}\bar{S}_{ik} > x) \leq c_i P(\bar{Z}_i > x).$$

We can now state the main result of this section.

Theorem 1 *Suppose that $\sum_{i=1}^{N} \rho_i < 1$ and that for some constant $\gamma > 2$, $s_i^{(\gamma)} < \infty$, $i = 1, \ldots, N$. Then, for all work-conserving nonpreemptive policies, we have that*

$$\lim_{k \to \infty} \sum_{i=1}^{N} \rho_i \left(\frac{\sum_{m=1}^{k} W_{im}}{k} \right) = V^* - \frac{1}{2}\sum_{i=1}^{N} \lambda_i s_i^{(2)} \quad a.e.,$$

where

$$V^* = \lim_{T \to \infty} \frac{1}{T} \int_0^T V^{(FIFO)}(t)dt = constant \quad a.e.$$

---

[1] Lemma 2 has been used in a different context in [4, 9].

Theorem 2 describes a hyperplane in the N-dimensional space, on which the limit of a linear combination of the Cesaro sums of response times must lie. Clearly, there are points in this hyperplane, that do not satisfy Theorem 2. The performance space (i.e., the subset of points that are achievable by some scheduling policy,) is well known for ergodic systems. For an $M|G|1$ system, we are able to characterize further the performance space, without the ergodicity assumption.

4 Limit Inequalities for $M|G|1$ systems.

In this section, we assume that the interarrival times are exponential. Then extensions to the inequalities that determine the performance space of ergodic work-conserving policies can be derived. As in [3, 2], the basic idea is to determine a lower bound on the time average of the work in system due to customers that belong to a subset R of the classes.

Suppose that a (work-conserving) policy π is in effect. Let $V_R^{(\pi)}(t)$ be the work in system at time t, due to customers that belong to the classes that are members of R, that is,

$$V_R^{(\pi)}(t) = \sum_{i \in R} V_i^{(\pi)}(t) \tag{5}$$

We are interested in developing a lower bound for

$$\liminf_{T \to \infty} \frac{\int_0^T V_R^{(\pi)}(t)dt}{T} \tag{6}$$

In [2], it was shown that for all $t \geq 0$ and for each sample path, $V_R^{(\pi)}(t)$ is minimized by a policy which gives nonpreemptive priority to classes in R. Moreover, since the order of service of customers from classes in R cannot affect $V_R^{(\pi)}(t)$, it is sufficient to assume that the order of service of customers from classes in R is FIFO. The question that needs to be resolved is whether the order of service of customers from R^c can affect the time average in (6). *Under the assumption that steady state exists,* it is shown in [3, 2] that the order of service of customers in R^c does not affect the time average in (6). Since we do not make any assumptions about the ergodicity of the policies, however, we need a different approach here.

Assume that a work-conserving policy π gives nonpreemptive priority to classes in R. Let τ_{im} be the time instant the m^{th} arriving customer from a class i in R^c is scheduled for service and let $\bar{\tau}_{im}$ be the first time in the interval $[\tau_{im} + S_{im}, \infty)$ at which there are no customers from classes in R in the system. Observe that $\bar{\tau}_{im} = \tau_{jl}$ for some $j \in R^c, l$, or $\bar{\tau}_{im}$ is the end of the kth busy cycle. Also, let τ_k be the first time within the k^{th} busy period that a customer from a class in R^c is scheduled for service. At time Γ_{k+1}, the integral in (6) can be written as follows:

$$\int_0^{\Gamma_{k+1}} V_R^{(\pi)}(t)dt = \sum_{m=1}^{k} \int_{\Gamma_m}^{\Gamma_{m+1}} V_R^{(\pi)}(t)dt = \sum_{m=1}^{k} \bar{U}_m^{(\pi)}, \tag{7}$$

where $\bar{U}_k^{(\pi)} \triangleq \int_{\Gamma_k}^{\Gamma_{k+1}} V_R^{(\pi)}(t)dt$. We may rewrite $\bar{U}_k^{(\pi)}$ in the form

$$\bar{U}_k^{(\pi)} = U_k^{(\pi)} + \sum_{i \in R^c} \sum_{m=M_{i(k-1)}+1}^{M_{ik}} U_{im}^{(\pi)}, \tag{8}$$

where

$$U_{im}^{(\pi)} = \int_{\tau_{im}}^{\bar{\tau}_{im}} V_R^{(\pi)}(t)dt \text{ and } U_k^{(\pi)} = \int_{\Gamma_k}^{\tau_k} V_R^{(\pi)}(t)dt \ .$$

We wish to show that $\{\bar{U}_k^{(\pi)}, k \geq 1\}$ is a sequence of i.i.d variables, the distribution of which does not depend on the scheduling policy. Recall that π is work-conserving, the arrival process is Poisson (and independent of the service requirements,) and the service requirements are independent random variables. Observe that the value of $V_R^{(\pi)}(t)$ for $t \in [\tau_{im}, \bar{\tau}_{im})$ depends only on S_{im} and the service times and the (Poisson) arrival process of customers from classes in R in that interval. It follows that the random variables $\{U_{im}^{(\pi)},\ i \in R,\ m \geq 1\}$ are independent and independent of $\{U_k^{(\pi)},\ k \geq 1\}$. Moreover, the distribution of U_{im} is independent of the the policy π. It is also clear that since π is nonidling and gives nonpreemptive priority to customers in R, $\{U_k^{(\pi)}, k \geq 1\}$ is a sequence of i.i.d. random variables and the distribution of $\{U_k^{(\pi)}, k \geq 1\}$ is independent of the policy. Observe also that the summation in (8) includes the *same* customers *independent* of the policy. This observation and the fact that $\{U_k^{(\pi)}, k \geq 1\}$ and $\{U_{im}^{(\pi)},\ i \in R^c,\ m \geq 1\}$ are independent sequences of independent random variables, imply that $\{\bar{U}_k^{(\pi)}, k \geq 1\}$ is a sequence of independent random variables and that the distribution of $\{\bar{U}_k^{(\pi)}, k \geq 1\}$ is the same for all work-conserving policies that give nonpreemptive priority to customers in R. Since under the policy that serves customers in R^c in $FIFO$ order the previous sequence consists of i.i.d. random variables, we finally have that

Proposition 1 *Under any work-conserving policy that gives nonpreemptive priority to customers in R, the sequence $\{\bar{U}_k^{(\pi)}, k \geq 1\}$ consists of i.i.d random variables identically distributed to the sequence $\{\bar{U}_k^{(\pi_f)},\ k \geq 1\}$ generated by the policy π_f that gives nonpreemptive priority to classes in R and serves customers from classes in R^c in FIFO order.*

Using Proposition 1 and assuming that the service discipline of customers from classes in R^c is FIFO, we conclude as in [3] that if classes in R have nonpreemptive priority, then under any nonidling policy,

$$\lim_{k\to\infty} \frac{\int_0^{\Gamma_k} V_R^{(\pi)}(t)dt}{\Gamma_k} = \frac{w_0 \sum_{i\in R}\rho_i}{1-\sum_{i\in R}\rho_i} + \sum_{i\in R}\rho_i s_i = F(R), \ a.e. \tag{9}$$

where w_0 is the residual work of the customer in service. Standard renewal arguments can be used now, to show that when classes in R have nonpreemptive priority,

$$\lim_{T\to\infty} \frac{\int_0^T V_R^{(\pi)}(t)dt}{T} = F(R), \ a.e. \tag{10}$$

From the above discussion, we conclude that

Proposition 2 *If $\sum_{i=1}^N \rho_i < 1$, then under any work-conserving policy π,*

$$\liminf_{T\to\infty} \frac{\int_0^T V_R^{(\pi)}(t)dt}{T} \geq F(R), \ a.e. \tag{11}$$

The limit exists and equality is achieved if classes in R have nonpreemptive priority over classes in R^c.

Finally, using Proposition 2 and the methodology of Theorem 1, we can show the following proposition.

Proposition 3 *Suppose that $\sum_{i=1}^{N} \rho_i < 1$, and for some constant $\gamma > 2$, we have $s_i^{(\gamma)} \leq \infty$, for $= 1, \ldots, N$. Then for any work-conserving nonpreemptive policy,*

$$\liminf_{k\to\infty} \sum_{i\in R} \rho_i \left(\frac{\sum_{l=1}^{k} W_{im}}{k} \right) \geq F(R), \quad R \subset \{1, \cdots, N\}. \tag{12}$$

The limit exists and equality is achieved, if classes in R have nonpreemptive priority over classes in R^c.

5 Conclusions.

We considered a $GI|G|1$ queueing system with N priority classes. The system operates under a scheduling policy which may be nonergodic or for which ergodicity may not be verifiable in advance. Adaptive scheduling rules represent an important class of such rules. We have derived an invariance relation that "conserves" a weighted sum of appropriate time (rather than statistical) averages. We have also demonstrated inequalities that characterize further the achievable performance region, for the special case of Poisson arrivals.

Acknowledgements

The work of the second author was supported by IBM, while he was a visiting scientist at the IBM T. J. Watson Research Center.

References

[1] R. Ash and M.F. Gardner. *Topics in Stochastic Processes.* Academic Press, New York, 1975.

[2] A. Federgruen and H. Groenevelt. $m/g/c$ queueing systems with multiple customer classes: characterization and control of achievable performance under nonpreemptive priority rules. *Management Science*, 34(9):1121–1138, 1988.

[3] E. Gelenbe and I. Mitrani. *Analysis and Synthesis of Computer Systems.* Academic Press, New York, 1980.

[4] L. Georgiadis. Relations between arrival and time averages of a process in discrete-time sytems with conditional lack of anticipation. Technical Report UVA/525415/EE87/102, Department of Electrical Engineering, University of Virginia, Charlottesville, Virginia, November 1986.

[5] L. Georgiadis, C. Nikolaou, and I. Viniotis. Adaptive scheduling algorithms that satisfy average response time objectives. In *Proceedings. Twenty-Seventh Annual Allerton Conf. on Comunication, Control, and Computing*, pages 810–919, Urbana, Champaign, September 27-29 1989. University of Illinois.

[6] L. Georgiadis and I. Viniotis. On the conservation law and the performance space of single server systems. technical report RC 15673, IBM T.J. Watson Research Center, 1990. Submitted for publication.

[7] P. Hall and C. C. Heyde. *Martingale Limit Theory and its Applications.* Academic Press, New York, 1980.

[8] D. P. Heyman and M. J. Sobel. *Stochastic Models in Operations Research*, volume I. McGraw-Hill, New York, 1982.

[9] A. Makowski, B. Melamed, and W. Whitt. On averages seen by arrivals in discrete time. In *Proceedings of the 28th IEEE Conference on Decision and Control*, December 1989. to appear.

[10] R.W. Wolff. Work-conserving priorities. *Journal of Applied Probability*, 7:327–337, 1970.

OPTIMAL ROUTING IN A SIMPLE NETWORK WITH DETERMINISTIC ARRIVAL AND SERVICE TIMES

M. Aicardi, R. Minciardi, R. Pesenti

Dipartimento di Informatica, Sistemistica e Telematica (DIST)
Via Opera Pia 11A, 16145 Genova

Abstract

The paper deals with the optimization of the mean flow time, over an infinite horizon, in a queueing system consisting of two heterogenous servers in parallel. The service time is deterministic for both servers, as well as the inter-arrival times of the single class customers. Some fundamental properties are first proved which have to be satisfied by infinite horizon optimal routing (dispatching) decision sequences. Then, an optimal (periodic) decision sequence is found, which besides corresponds to a stationary stabilizing assignment policy.

1. Introduction and background

Optimal control of queueing systems has received an increasing attention in recent years (for a survey of early work in this field, one can see Stidham and Prabhu [1]). Nevertheless, one may still complain about the lack of general results and powerful methodologies, at least in comparison with optimal control of "conventional" dynamical systems.

In this context, a class of basic and conceptually simple problems have been studied, which are related to customer assignment to two or more servers. Actually, this can be viewed as the "elementary" dynamic decision problem in a queueing network. In this connection two different decisional models have been considered.

The first of these is related to the assignment of customers belonging to a single queue. For such a problem and for the case of two servers, Lin and Kumar [2] and Walrand [3] have determined the optimal dynamical strategies minimizing the discounted or average number of customers in the system, in the assumption of Poisson arrivals and exponential service times.

The optimal policy turns out to be of "threshold" type (i.e., the fast server should be kept busy whenever possible and the slow server should be used only when the queue length overcomes a certain threshold). Successively, Viniotis and Ephremides [4] considered an extension of such a problem in case one of the two servers has exponential, and the other has erlangian distribution, and the customer arrival distribution is of general type. Even in such assumptions the optimality (or quasi-optimality) of threshold type policy is proved.

A second decisional model arises when assignment decisions are taken immediately upon the arrival of customers. In other words in this case each server has its own queue. This model has been cosidered by Winston [5] in case of Poisson arrivals and exponential service times. Within such assumptions, the optimality of the shortest line discipline is proved, when the discounted number of customers in the system is to be minimized.

With reference to the same decisional model, a different optimization objective is considered by Ephremides et al. [6], concerning the total completion time of jobs (customers) arrived in a certain time interval (with arrival times known in advance). The specific model is, in this case, made up by two identical exponential servers. It is shown that if the queue lengths are observed, then the best policy is to route jobs to the shortest queue, whereas, if the queue lengths are not observed, it is best to alternate between queues.

However, to the best of the authors' knowledge, no general result about the second of the considered simple decisional models has ever appeared in the literature. More specifically, the optimal routing (assignment) of incoming customers to downstream parallel queues, within general assumptions about arrival and service times distribution, is still an open problem.

In this paper, we consider a decisional problem falling in the second of the considered classes. The main feature of our problem is that it is completely deterministic, both in service times and customers arrivals. Our objective is the minimization of the mean flow time of customers.

2. The model and the optimization problem

We consider a system with two server, each one with its own queue. Each server M_i follows a first-come-first-served discipline, and, at each time instant, the system state can be represented by

$$x(t) = [w_1(t), w_2(t)]$$

where $w_i(t)$ is the emptying time of the server M_i, taking into account its present queue.

A single class of identical customer is present in the system, whose arrival process is entirely deterministic. Namely, $T = \frac{1}{\lambda}$ is the constant interarrival time of the customers. In the same way, also the service process is deterministic. Namely, $S_i = \frac{1}{\mu_i}$ is the constant service time of server M_i.

We assume the following inequalities hold:

$$S_2 > S_1 > T \tag{1}$$

$$\frac{1}{T} \leq \frac{1}{S_1} + \frac{1}{S_2} \quad \text{(stabilizability condition)} \tag{2}$$

Without significant loss of generality, we assume that T, S_2, S_1, are integer. If the equality (2) holds, then we say that the system (characterized by S_2, S_1) is on the frontier of the stabilizability region.

As already specified, we assume that the customers arrive (and are instantaneously assigned to one of the two queues) at time instants kT, k=0,1,2,.... Let $u_i(kT)$ the decision variable equal to 1 if the k-th customer is assigned to M_i, and zero otherwise.Obviously, it must be $u_1(kT) + u_2(kT) = 1$. Then the state equation of each subsystem {waiting line plus server} is:

$$w_i(kT) = \max(w_i((k-1)T) - T, 0) + u_i(kT)S_i \qquad i=1,2 \tag{3}$$

where $w_i(kT)$ is the empyting time of server M_i after the assignment of the customer arriving at time kT (i.e., the (k+1)th customer).

The set Π of assignment (control) policies of interest is that of non-randomized stationary policies functions of the system state, namely policies of the type

$$u(kT) = \pi[x(kT^-)] \qquad k = 0,1,2,\ldots \tag{4}$$

being $u(kT) = [u_1(kT), u_2(kT)]$, and $x(kT^-)$ the system state immediately before the (k+1)th assignment. Let us define Π_S, subset of Π, as the set of stabilizing policies, i.e., policies which, indipendently of the initial state x(0), do not allow the system state to grow indefinitely. It is plain to understand that, under the assumed conditions about our model, the set Π_S is certainly not empty.

Clearly, due to the integrity assumption of T, S_2, S_1, every policy belonging to Π_S eventually makes the system periodically visit a certain number of states, in a definite order. The sequence of states, periodically visited is, in general, dependent on the initial state, for a given policy $\pi \in \Pi_S$. In addition, there is the possibility of a transient behaviour, before the system reaches one of the states of the periodic sequence. This transient behaviour depends on the initial state, too. Of course, sequences of assignments correspond to the above mentioned sequences of states; with a little abuse of the terminology, we will call such sequences of assignments "periodic", even though their initial part can correspond to a transient behaviour.

With respect to policies belonging to Π_S, and in the assumption of zero initial state (i.e., $x(0) = (0,0)$), we want to minimize the mean flow time of customers over infinite horizon, i.e., the cost functional

$$\overline{F} = \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} \sum_{i=1}^{2} w_i(kT)\, u_i(kT) \tag{5}$$

3. Determination of the optimal policy

In this section, some results about the solution of the previously stated optimization problem will be given.

As previously mentioned, the application of any stationary policy eventually makes the system state cyclically visit a certain number of different states, which in turn implies that a fixed cycle of assignments decisions takes place. In the following, we will initially deal only with periodic sequences of assignments and will provide some results about the possibility of obtaining better periodic (and stabilizing) assignment sequences. It is worth mentioning that, as it is easy to show,a periodic and stabilizing sequence of assignments generates a periodic sequence of states. Remember that we use the attribute "periodic" in an enlarged sense, since we admit the possibility of an initial transient behaviour, before the properly periodic sequence.

Let us begin with some preliminary results.

Result 1

Consider a given periodic sequence of assignments (call it "nominal") and the associated periodic sequence of states. Let P be the m.c.m. between the two periods of the sequences. Then, neglect the initial transient, since it cannot have any influence on the considered cost functional (over infinite horizon). Consider a subsequence of states (and assignments) corresponding to the first period. Suppose that in such subsequence a certain number of time instants, say $k_1T, \ldots, k_sT$, exist such that:

i) $w_m(kT^-) \geq T$ and $w_n(kT^-) < T$.

ii) $J_{k_i} \to M_m$ (i.e., customer J_{k_i} is assigned to server M_m); $J_{k_i+1} \to M_n$.

Then, a "new" periodic sequence (with the same initial transient as the nominal one) can be built up by means of the following procedure:

a) set $i = 1$;

b) test whether, at time instant k_iT, conditions i) and ii) still hold; if not skip to d; otherwise continue;

c) interchange the assignments at time instants k_iT and $(k_i+1)T$, i.e. set $J_{k_i} \to M_n$ $J_{k_i+1} \to M_m$, and at any couple of time instants $[(k_i+rP)T,(k_i+1+rP)T]$, $r \geq 1$;

d) if $i = s$ stop; otherwise set $i = i+1$ and go to b.

For the considered optimization objective, the obtained new periodic sequence is no worse than the nominal one.

▲

Proof

We can consider first the time instants k_1T and $(k_1+1)T$. Let us show that interchanging the assignments made in the nominal sequence at such time instants, we obtain a new sequence not worse than the previous one.

In fact, the flow time of customers arrived before J_k, cannot change. Moreover the flow times of J_{k_1} and J_{k_1+1}, in the nominal sequence are $F(J_{k_1}) = w_m(k_1T^-) + S_m$ and $F(J_{k_1}+1) = \max[w_n(kT^-) - T,0] + S_n$, respectively. After the assignment interchange, instead, we have $F'(J_{k_1}) = w_n(k_1T^-) + S_n$ and $F'(J_{k_1}+1) = w_m(kT^-) - T + S_m$. Then, globally, the aggregate flow time of such pair of tasks cannot grow worse. Finally, the flow times of customers which arrive after J_{k_1+1} cannot grow worse, due to the considered interchange. In fact the flow times of such customers assigned to M_m does not change, whereas those of customers which are assigned to M_n can only diminish or remain the same.

At this point, since w_m remains unchanged, whereas w_n does not increase for any instant $t \geq (k_1+2)T$, surely the two conditions i) and ii) still hold (after the above assignment interchange) at the time instants $[(k_1+P)T,(k_1+1+P)T]$. Thus we can again perform an assignment interchange, again not worsening the sequence. Since the same reasoning line can be used at any pair of instants $[(k_1+rP)T,(k_1+1+rP)T]$, $r \geq 1$, we can obtain eventually a periodic sequence not worse than that before considered.

Then we pass to time instants k_2T, $(k_2+1)T$ and check whether condition i), ii) still hold; in the affermative case, we perform the interchange assignments for such time instants, and for all pair of time instants$[(k_2+rP)T,(k_2+1+rP)T]$, $r \geq 1$. The procedure continues until no time instants exist in the period where conditions i) and ii) are true. Since none of the introduced sequence modification can worsen the sequence, the result is proved.

■

By repeated use of Result 1, a simple argument leads to consider only periodic sequences of assignments which satisfy the following properties:

i) $w_1(kT^-) \geq T \Rightarrow J_k \rightarrow M_2$
ii) $w_2(kT^-) \geq T;\ w_1(kT^-) < T \Rightarrow J_k \rightarrow M_1$.

Periodic sequences satisfying such implications will be said to belong to a set Σ_a.

Result 2

It can be shown that (starting from the zero initial state) the application of any periodic sequence of decisions $\sigma \in \Sigma_a$ implies that at every time instant kT, at least one of the following conditions holds:

$w_1(kT^-) < T$
$w_2(kT^-) < T$.

▲

Proof

We proof the result ab absurdo. That is, we first derive a consequence of what is denied by the result, and then we prove that this consequence cannot be true, by itself.

Then, let us show that, if we assume that, at a certain time instant, say sT, we have $w_1(sT^-), w_2(sT^-) \geq T$, that implies that a time instant rT exists $(r < s)$ such that:

i) $w_1(rT^-), w_2(rT^-) < T$;
ii) none of the two serves is idle between time instant (r+1) and time instant sT.

In order to prove this implication, we make use of what follows from Result 1. The existence of a time instant satisfying i) is trivially ensured by the zero initial state. Let us call rT the last of such time instant before sT, and consider what happens from time instant rT to sT. If some time instant mT ($r < m < s$) exists such that one of the servers, say M_j, has $w_j(mT^-) < T$, then two situations can occur: either $w_k(mT^-) < T$ ($k \neq j$), or $w_k(mT^-) \geq T$ (and then $J_m \rightarrow M_j$ since only sequences belonging to Σ_a are considered). In the first case, time instant mT satisfies ii) and then rT could not be the last time instant of interest. In the second case both machine are not idle between mT and (m+1)T. This proves that, since when the last time instant satisfying i) is encountered, up to time instant sT, no machine can become idle.

Thus, let us prove that the following fact cannot take place: a time instant sT exists, with $w_1(sT^-)$, $w_2(sT^-) \geq T$, which implies (due to what has just been shown above) the existence of a time instant rT, $r < s$, with $w_1(rT^-)$, $w_2(rT^-) < T$, and M_1,M_2 idle between (r+1)T and sT.

In fact, let us consider how many customers are completely served in the considered time interval. Due to the fact that machines are never idle between (r+1)T and sT, this number is

$$n(r+1,s) = \left\lfloor \frac{(s-r-1)T}{S_1} \right\rfloor + \left\lfloor \frac{(s-r-1)T}{S_2} \right\rfloor \tag{6}$$

Due to the stabilizability hypothesis (2) we have

$$\frac{(s-r-1)T}{S_1} + \frac{(s-r-1)T}{S_2} \geq s-r-1 \tag{7}$$

Then, since (s-r-1) is integer, this implies that

$$n(r+1,s) \geq s-r-2 \tag{8}$$

Observe now that the total number of customers present in the system in $(r+1)T^-$ is one, whereas (s-1-r) customers enter in the system from time instant (r+1)T up to time instant (s-1)T. Then, since at least (s-r-2) have left the system in such time interval, the total number of customers present at time instant sT^- cannot exceed two. Since $w_1(sT^-)$, $w_2(sT^-) \geq T$, the customers present in the system at sT^- must be exactly two. A third customer enters at time instant sT. By computing via relations analogous to (6) (7) and (8) the number of customers completed over the interval [(r+1)T, (s+1)T] , it comes out that at time instant $(s+1)T^-$, the total number of present customer cannot exceed two, again. This shows that one customer must leave the system between sT and $(s+1)T^-$, thus implying the impossibility of having both $w_1(sT^-)$, $w_2(sT^-) \geq T$.

■

Result 3

Let a given sequence $\sigma \in \Sigma_a$ yield a certain number of "clusters" of consecutive customer assignments to M_2. Then, it is always possible to derive a different periodic sequence σ' characterized by:

i) there is no consecutive assignment to M_2;

ii) the mean flow time associated with σ' is not greater than the one associated with σ.

▲

Proof

The proof is trivial whenever $S_2 \geq 2T$. In this case, sequence belonging to Σ_a cannot perform two consecutive assignments to M_2, due to result 2.

The proof is less trivial whenever $S_2 < 2T$. In this case we prove the statement in three steps.

Step 1.

We prove that if two or more consecutive customers are assigned to M_2, then, assuming that the assignment sequence is optimal, at the time instant kT when the last customer has been assigned to M_1 the following relation must hold

$$w_1(kT^-) + S_1 < w_2(kT^-) + S_2 \tag{9}$$

In fact, consider the following three assignments, $J_k \rightarrow M_1$, $J_{k+1} \rightarrow M_2$, $J_{k+2} \rightarrow M_2$, and assume that the previous relation (9) is not true. Then, the considered three assignments can be modified without worsening the sequence performance (needless to say, since we are considering periodic sequences, the same argument as in the Proof of Result 1 can be used to repeat the considered modification in order to have a new periodic sequence; moreover, it is plain to see that this modification yields a sequence belonging to Σ_a).

Actually, consider the sequence obtained through the following modification: $J_k \rightarrow M_2$, $J_{k+1} \rightarrow M_1$, $J_{k+2} \rightarrow M_2$. By the way, it is easy to show that, assuming (9) is not true, it necessarily turns out $J_{k-1} \rightarrow M_1$, so that the introduced modification does not increase the number of consecutive assignments to M_2.

Let us compute the overall flow time of the first two of the considered three customers in the unmodified sequence

$$F = w_1(kT^-) + S_1 + \max(w_2(kT^-) - T,0) + S_2$$

and in the modified sequence

$$F' = w_2(kT^-) + S_2 + \max(w_1(kT^-) - T,0) + S_1$$

To prove $F \geq F'$ in all possible cases, it is sufficient to note that if (9) is not true this implies $w_1(kT^-) > w_2(kT^-)$.

Further, let us consider the flow time of customers entering after J_{k+1}. Obviously the flow time of customers assigned to M_2 cannot increase by the introduced modification. Instead, the flow times of customers assigned to M_1 remain exactly the same, since $w_1((k+3)T^-)$ is not affected by the modification. In fact before the modification

$$w_1((k+3)T^-) = \max(w_1(kT^-) + S_1 - 3T,0)$$

and after it

$$w_1((k+3)T^-) = \max(\max(w_1(kT^-) - T,0) + S_1 - 2T,0) =$$
$$= \max(w_1(kT^-) - T + S_1 - 2T, S_1 - 2T,0)$$

which is identical to the above expressed by taking into account that $S_1 < 2T$, due to the assumptions (1), (2). This concludes step 1 of the proof.

Step 2.

We prove that, in the assumption of a sequence of consecutive assignments to M_2, say $J_{k+1} \rightarrow M_2$, $J_{k+2} \rightarrow M_2$, ..., $J_{k+s} \rightarrow M_2$, then for every time instant $(k+r)T$, $2 \leq r \leq s$, it follows

$$w_1((k+r)T^-) \leq w_2((k+r)T^-) \tag{10}$$

In fact, let us compute the state variables in time instants (k+1)T, (k+2)T,.... We have:

$$w_1((k+1)T^-) = w_1(kT^-) + S_1 - T \qquad w_2((k+1)T^-) = \max(w_2(kT^-) - T, 0)$$
$$w_1((k+2)T^-) = \max(w_1(kT^-) + S_1 - 2T, 0) \qquad w_2((k+2)T^-) = \max(w_2(kT^-) - T, 0) + S_2 - T$$
$$\ldots \quad \ldots \qquad\qquad \ldots \quad \ldots$$

Then, by taking into account (9), that must be true since we are looking for optimal sequences, (10) is proved.

Step 3.

We now want to show, that, in presence of a series of consecutive assignments to M_2, as considered at the beginning of step 2, we can improve the periodic assignment sequence, by introducing a suitable modification. The same remark as in step 1 applies about the repeatability of the modification and the belonging of the resulting sequence to Σ_a.

Let us consider the last two successive assignments to M_2 in a period before two successive assignments to M_1(needless to say, such situation must necessarily occur otherwise there would be proportionally more jobs assigned to M_2 than to M_1; moreover this operation can be repeated until there are two consecutive assignments to M_2 in the period). The general situation can be depicted as

	$(k+s)T$	$(k+s+1)T$	$(k+s+2)T$		$(k+s+2+2n)T$
customers	... ... J_{k+s-1}	J_{k+s}	J_{k+s+1}	... ...	$J_{k+s+2+2n}$ ...
assignments	... ... M_2	M_2	M_1	$\{M_2\ M_1\}^n$	M_1 ...

where $\{M_2\ M_1\}^n$ $(n \geq 0)$ denote n pairs of alternate assignments. As it will be shown the following assignment modification

	$(k+s)T$	$(k+s+1)T$	$(k+s+2)T$		$(k+s+2+2n)T$
customers	... ... J_{k+s-1}	J_{k+s}	J_{k+s+1}	... ...	$J_{k+s+2+2n}$ ...
assignments	... ... M_2	M_1	M_2	$\{M_1\ M_2\}^n$	M_1 ...

does not worsen the cost functional,. Obviously the modified sequence does not change the number of customers assigned to each machine. Consider first the flow times of customer assigned over the interval $[(k+s)T, (k+s+2+2n-1)T]$ (i.e. up to the last customer assigned to M_2 in the modified sequence). For the original sequence the partial sum of the flow times can be written as

$$F = \sum_{i=0}^{n} ((\eta_i + S_2) + S_1) \tag{11}$$

since

a) being $w_1((k+s)T^-) < T$ and $S_1 < 2T$, each customer asigned to M_1 has a flow time equal to S_1.

b) each customer assigned to M_2 has flow time equal to $\eta_i + S_2$, where η_i represents the waiting time of every job assigned to M_2 over the considered interval.

Actually

$$\begin{cases}\eta_{i+1} = \max(\eta_i + S_2 - 2T, 0)\\ \eta_0 = w_2((k+s)T^-)\end{cases}$$

As to the modified sequence, by using the same reasoning line we get

$$F' = \sum_{i=0}^{n} ((\zeta_i + S_1) + S_2) \tag{12}$$

where now ζ_i represents the waiting time of every job assigned to M_1 and is given by

$$\begin{cases}\zeta_{i+1} = \max(\zeta_i + S_1 - 2T, 0)\\ \zeta_0 = w_1((k+s)T^-)\end{cases}$$

Now note that $S_1 < S_2$ and $w_1((k+s)T^-) \leq w_2((k+s)T^-)$ together imply $\zeta_i \leq \eta_i \ \forall i$ and then $F' \leq F$.

Then, to complete the proof, it is sufficient to show that the overall flow time of all the successive customers does not increase. This is equivalent to saying that the waiting times corresponding to the first assignment to each server (after time instant $(k+s+2+2n-1)T$) are, in the modified sequence, no worse than in the original one. In fact in the new situation we have $w_2(((k+s+2+2n+1)T^-)=0$ whereas $w_1(((k+s+2+2n)T^-) \leq S_1 - T$, being $S_1 - T$ the waiting time of $J_{k+s+2+2n}$ on M_1 in the original sequence.

■

Moreover we are able to bound the number of consecutive assignments to M_1, between two assignments to M_2, as the following corollary will explain.

Corollary 1

As it can be easily derived from the previous Results, the number 'a' of any group of consecutive assignments to M_1 satisfies the following inequalities:

$$\frac{\varepsilon_2}{T} - 1 < a < \frac{T}{\varepsilon_1} + 1 \tag{13}$$

where $\varepsilon_i = S_i - T$, $i=1,2$.

▲

Proof

Since in any time instant kT in which a customers J_k is assigned to M_1, $w_1(kT^-) < T$ then, if 'a' consecutive customers are assigned to M_1, it follows that $w_1((k-a)T^-)+(a-1)\varepsilon_1 < T \Rightarrow (a-1)\varepsilon_1 < T$. Similarly $J_k \rightarrow M_2$ implies $w_2((k-a)T^-)+\varepsilon_2 - aT < T \Rightarrow (a+1)\varepsilon_2 > T$.

■

Due to Result 3, we can restrict our attention to periodic assignment sequences that, after the possible transient part, can be characterized by the repetition of the assignment cycle $B \equiv ((a_1,b_1),(a_2,b_2),\ldots,(a_n,b_n)) \equiv (\{a_i,b_i\}^n)$, for some n, where a_i is a number of consecutive assignments to M_1 and $b_i=1 \ \forall i$ is a single assignment to M_2.

We are now ready to provide the two basic results.

Theorem 1

a) If an integer a° exists satisfying conditions

$$\frac{\varepsilon 2}{T} \leq a° \leq \frac{T}{\varepsilon 1} \tag{14}$$

and

$$\sqrt{\frac{1}{4}+\frac{2\varepsilon 2}{\varepsilon 1}}-\frac{3}{2} \leq a° \leq \sqrt{\frac{1}{4}+\frac{2\varepsilon 2}{\varepsilon 1}}-\frac{1}{2} \tag{15}$$

(actually, constraint (15) can be satisfied at most by two integers and at least by one), then the periodic assignment sequence corresponding to B≡(a°,1), with no initial transient, cannot be improved.

b) Otherwise, i.e., if (14) and (15) have no common integer solution then there exists a periodic assignment sequence (again with no initial transient), having the structure below indicated, that cannot be improved. This structure can be represented by $B \equiv (\{a_i,1\}^n)$ where $a_i=a°$ or $a=a°+1$, being a° an integer and

$$\frac{\varepsilon 2}{T}-1 < a° < a°+1 < \frac{T}{\varepsilon 1}+1 \text{ ,}$$

and B is given, for some r, by B_r recursively defined as follows:

$$B_1(k_1) \equiv (\{a°,1\}^{k_1},(a°+1,1)) \text{ or } (\{a°+1,1\}^{k_1},(a°,1))$$

$$B_2(k_1,k_2) \equiv (\{B_1(k_1)\}^{k_2},B_1(k_1+1)) \text{ or } (\{B_1(k_1+1)\}^{k_2},B_1(k_1))$$

$$B_r(k_1,\ldots,k_r) \equiv (\{B_{r-1}(k_1,\ldots,k_{r-1})\}^{k_r},B_{r-1}(k_1,\ldots,k_{r-1}+1))$$
$$\text{or } (\{B_{r-1}(k_1,\ldots,k_{r-1}+1)\}^{k_r},B_{r-1}(k_1,\ldots,k_{r-1}))$$

A simple algorithm can be devised to determine exactly a° and $k_1,\ldots,k_r$ and so the optimal periodic sequence.

▲

Proof (sketch)

Let us consider case a) and the finite sequence of assignment (a,1) which has the minimum flow time, given $w_1(0) = w_2(0) = 0$. The optimal value of 'a' is given by the integer which satifies

$$\sqrt{\frac{1}{4}+\frac{2\varepsilon 2}{\varepsilon 1}}-\frac{3}{2} \leq \bar{a} \leq \sqrt{\frac{1}{4}+\frac{2\varepsilon 2}{\varepsilon 1}}-\frac{1}{2}$$

If $\bar{a}$ satisfies also (14) than the periodic sequence given by the repetition of this subsequence is optimal. In fact, starting from zero initial conditions all the customers assigned to a given machine by some subsequence $(\bar{a},1)$ are completely served before the first customer assigned by the succeding subsequence $(\bar{a},1)$ to the same machine. The same argument can be used for successive subsequences. Moreover, no other repeatable subsequence has a mean flow time lower than that determined.

As to case b) it can be proved that

$$\text{if } \bar{a} > \frac{T}{\varepsilon 1} \ (\bar{a} < \frac{\varepsilon 2}{T}) \text{ then } a° = \left\lfloor \frac{T}{\varepsilon 1} \right\rfloor \ (a° = \left\lfloor \frac{\varepsilon 2}{T} \right\rfloor).$$

■

It can be shown that the optimal periodic assignment sequences that come out either in case a) or in case b) in Theorem 1, can be obtained, in the steady state, by the application of some stationary stabilizing policy $\pi \in \Pi_S$. In particular, the optimal policy is a constant threshold policy based on the difference between the emptying time of the two queues. Formally, we have

Theorem 2

$\exists\ \lambda$ such that the application of

$$u_1(kT) = \begin{cases} 0 & \text{if } w_1(kT^-) - w_2(kT^-) \geq \lambda \\ 1 & \text{otherwise} \end{cases}$$

generates, in the steady state, the optimal assignment sequence, that cycles through the same states visited by the application (from t=0) of the optimal periodic sequence of assignment, defined by Theorem 1.

▲

The proof of these theorems, which are somewhat lengthy and tedious are omitted here for sake of brevity. They will be reported in a subsequent paper.

4. Conclusions

The purpose of this paper has been that of providing some basic results about a simple, yet unsolved to the best of the authors' knowledge, optimization problem over infinite horizon. Actually, despite of the simplicity of the statement of the problem, the way to derive the optimal assignment policy has been rather cumbersome. Although the main results of the paper have not been proved, due to the lack of space, we assume to have shown the basic arguments and results concerning the optimality of the assignment sequence (and of policies). Further research will be devoted to face optimization problems corresponding to more complex systems and to explore the connections between optimization problems over finite and infinite horizon.

References

[1] S. Stidham, N. Prabhu, "Optimal control of queueing system", in Math. Methods in Queueing Theory, A.B. Clarke, Ed. , Lecture Notes in Econom. and Math. Syst., vol.98, Spinger Verlag, New York, pp 263-294, 1974.

[2] W. Lin, P.R. Kumar, "Optimal control of queueing system with two heterogeneous servers", IEEE Trans. Autom. Contr., vol. 29, pp.696-703, 1984.

[3] J. Walrand, "A note on 'Optimal control of queueing system with two heterogeneous servers'", Syst. & Contr. Lett., vol.4, pp.133-134, 1984.

[4] I. Viniotis, A. Ephremides, "Extension of the optimality of the threshold policy in heterogeneous multiserver queueing systems", IEEE Trans. on Autom. Contr., vol. 33, pp.104-109, 1988.

[5] W. Winston, "Optimality of the shortest lime discipline", J. Appl. Prob., vol. 14, pp 181-189, 1977.

[6] A. Ephremides, P. Varaiya, J. Walrand, "A Symple Dynamic Routing Problem", IEEE Trans. Autom. Contr., vol. 25, pp.690-693, 1980.

A Fast Stochastic Approximation Algorithm for Optimization of Dynamical Systems

Hung-Jen Chang
Kadir Liano

The University of Texas at Austin
Department of Electrical and Computer Engineering
Austin, Texas 78712-1084

Abstract

Most recursive approximation methods rely only on the most recent estimation of the function. This study proposes a method which utilizes both present and past data to optimize the performance of Discrete Event Dynamical Systems (DEDS). The advantages of this approach in terms of the accuracy of results and convergence rates are shown empirically on simulation runs of M/M/1 queues.

1 Introduction

From engineering, economical, and other points of view, it is always desirable to have a system operating at its optimum point. However, there are many complex systems where the relation between the objective function and the system parameters are not known and thus cannot be optimized analytically. The dependency on system parameters can only be obtained by either running simulation model or from observing the actual system. Computer simulation is often preferred for various reasons: the actual system is not easily available or accessible, or changing the parameters of an actual system is very costly. Due to the stochastic nature of the system, one sample estimate obtained from a single simulation run is usually not reliable. In order to obtain an accurate estimate of the optimum point, a large number of simulation runs are performed to reduce the effect of noise on the system.

A new stochastic optimization method with faster convergence is proposed. Assuming that the gradient of the objective function is non decreasing, optimization is equivalent to finding the zero crossing of the gradient. The gradient of the system response function with respect to the system parameters can be estimated using the Infinitesimal Perturbation Analysis (IPA) algorithm which is reviewed in Section 2. Details of the proposed Quadratic Least Square (QLS) algorithm are described in Section 3. Empirical study in Section 4 compares the analytical solutions of an M/M/1 queue to the simulation results of the proposed method. These results are also compared to the recently published results by Suri and Leung [5]. Section 5 concludes this study.

2 Infinitesimal Perturbation Analysis

Perturbation Analysis (PA) originated by Ho [3] enables the estimation of the gradient of the system response function with respect to system parameters from a single sample path. This technique is applicable on any Discrete Event Dynamic System (DEDS). Infinitesimal Perturbation Analysis (IPA) which falls within the general class of PA techniques is used on systems with continuous parameters. The IPA algorithm is summarized below for a single server queue with one system parameter and one system response.

Let the system parameter θ be the mean service time, the system response $T(\theta)$ be the mean system time, T_θ be the first derivative of T with respect to θ, and x_j be the actual service time of customer *j*.

```
Initialize
    $sumdx = 0$
    $sumdT = 0$

At the end of service for each customer $j$
    $sumdx = sumdx + \frac{dx_j}{d\theta}$
    $sumdT = sumdT + sumdx$
    If the server is idle then $sumdx = 0$
    Repeat these steps for L customers

After L customers are served
    $T_\theta = \frac{sumdT}{L}$
```

This algorithm produces an estimate of the gradient of the mean system response time with respect to the mean service time. Detailed explanation of this technique can be found in other papers in the reference and is not repeated here.

3 Quadratic Least Square (QLS) Algorithm

One important feature of the proposed algorithm that is missing from traditional hill climbing methods is the global approach in finding the optimum solution. Iterative hill climbing algorithms use only local information at present point in calculating the gradient. Knowledge about the entire surface is not retained since all previous information are discarded. Instead of going down the hill step by step on the surface of the objective function, the proposed algorithm attempts to fit a quadratic surface on it and therefore can directly locate the optimum point. This algorithm is efficient in the sense that information collected at all previous points are utilized in finding the best fit. Since quadratic surface is a good fit locally to any smooth surface, the accuracy of the solution can be improved by forgetting old points as the algorithm home in on the optimum point.

The general form of the objective function J of a DEDS and its derivative with respect to the system parameters can be written as follow.

$$J = J(\theta,F(\theta))$$

$$J_\theta = \frac{dJ}{d\theta} = \frac{\partial J}{\partial \theta} + \frac{\partial J}{\partial F}\frac{dF}{d\theta} = \frac{\partial J}{\partial \theta} + \frac{\partial J}{\partial F}F_\theta$$

The objective function J is known in term of the system parameters θ and the system responses $F(\theta)$. However, finding the optimum value of J with respect to system parameters is difficult since the exact form of $F(\theta)$ is usually not known. On the other hand, $\frac{\partial J}{\partial \theta}$ and $\frac{\partial J}{\partial F}$ can be derived easily from J; and F_θ can be estimated using IPA technique described earlier. Therefore the zero crossing of J_θ which is equivalent to the optimum point of the objective function can be computed.

The proposed algorithm uses the fact that for a small enough interval, any continuous curve, J_θ in this case, can be closely estimated by a straight line. A least square fit line is calculated based on all the existing points. The system parameters is calculated from the zero crossing of this line and the derivative of the objective function is then computed. This provides a new data point for the least square fit. The procedure is repeated until the zero crossing at subsequent iterations are fixed within an acceptable tolerance. Although it is desirable to use as many data points as are available, it should be noted that the linear approximation of a curve is only good on a small interval. Since it is expected that the most recently calculated points are more accurate, a running window with a fixed length is used to discard older points.

A step by step listing of the proposed algorithm is given below.

#1 Fix the window size and select three θ's for the starting points
#2 Run IPA to estimate $J_\theta|_{\theta=\theta_i}$
#3 Fit a least square line $J_\theta = a\theta + b$
#4 Compute $\theta_i = \alpha\theta_{i-1} - (1-\alpha)\left(\frac{b}{a}\right)$
#5 If ($|\theta_i - \theta_{i-1}| >$ ERR) then go to #2
#6 Done

Note that the window size specified in #1 controls the number of points used in the least square fit in #3. At the first few iterations, data points are added to the window. When the window is filled up, older data are discarded and replaced by the new data. The number of data points are kept constant from then on. Since the newer data are clustered near the optimal point, the linear approximation of J_θ is expected to be quite accurate.

The zero crossing of the linear equation in #3 is located by setting J_θ to zero. This occurs at $\theta = -\frac{b}{a}$. Note that the values of *b* and *a* are computed from all data inside the window. In other words a global update is attempted in each step as opposed to the local gradient updates in the hill climbing methods. The coefficient α used in #4 is often referred to as the momentum which takes on value between zero and one. This is a common technique used in many recursive numerical algorithms to minimize oscillation and to improve the convergence. The algorithm uses a simple stopping criteria as shown in #5. It assumes convergence when two consecutive estimates of the optimum point are close to each other. The last value of θ_i is the estimated value of the optimum point.

4 Empirical Study

The QLS algorithm is used to optimize a single server M/M/1 queue with respect to the mean service time in the following empirical study. Since this system is rather simple, there exists analytical formula to evaluate the system performance. This analytical solution can then be used as a standard to measure the performance of the algorithm.

4.1 M/M/1 Queue

The notations used in previous sections are listed again for convenience. The objective function of the system is defined and the optimal solutions are derived.

θ = mean service time
θ^* = optimum mean service time
$T(\theta)$ = mean system time
ρ = utilization

ρ^* = utilization when system is optimum
λ = mean arrival rate
C_0 = cost per unit time in the system
C_1 = cost per unit speed of the server
$J(\theta,T)$ = objective function

The objective function reflects the trade off between the cost in term of the delay a customer experiences in the system and the cost of having a faster server. It is defined below along with the derivation of other optimum functions.

$$J = C_0 T(\theta) + \frac{C_1}{\theta}$$

$$\frac{dJ}{d\theta} = C_0 T_\theta(\theta) - \frac{C_1}{\theta^2}$$

For an M/M/1 queue,

$$T(\theta) = \frac{\theta}{1 - \lambda\,\theta}$$

$$T_\theta(\theta) = \frac{1}{1 - \lambda\,\theta} + \frac{\lambda\theta}{(1 - \lambda\theta)^2}$$

$$\frac{dJ}{d\theta} = C_0\left[\frac{1}{1 - \lambda\,\theta} + \frac{\lambda\theta}{(1 - \lambda\theta)^2}\right] - \frac{C_1}{\theta^2}$$

Set $\frac{dJ}{d\theta} = 0$ to calculate θ^*

$$\theta^* = \frac{\sqrt{C_1}}{\sqrt{C_0} + \lambda\sqrt{C_1}}$$

$$\left.\frac{dJ}{d\theta}\right|_{\theta=\theta^*} = \sqrt{C_0 C_1} - \frac{C_1}{\theta^2}$$

$$T(\theta^*) = \frac{\sqrt{C_1}}{\sqrt{C_0}}$$

$$J(\theta^*) = 2\sqrt{C_0 C_1} + \lambda C_1$$

Notice that the mean service time θ is a positive number and the utilization $\rho = \lambda\theta$ which is defined as the ratio of busy time over observed time is always bounded between zero and one. These imply that the range of θ is $0 < \theta < \frac{1}{\lambda}$. This bound is used later in the simulation to provide the initial value of the mean service time. To enable comparison with previous works, several system parameters in the simulation are set to some specific values: $\lambda = 0.01$, $C_0 = 1$, and the values of C_1's are chosen such that the optima occur over a wide range of utilizations. Table 1 provides a list of the values for C_1 with the corresponding optimum mean service time and optimum utilization.

C_1	θ^*	$\rho^* = \lambda\theta^*$
2000	30.90	0.31
8000	47.21	0.47
32000	64.14	0.64
128000	78.16	0.78
512000	87.74	0.88

Table 1. Analytical optimal service time.

4.2 Implementation Details

Besides the system parameters which values are specified above, additional parameters are required for the simulation. Since the objective function is estimated by a second order polynomial, three initial guesses of the mean service time θ are specified. It has been shown above that the optimal mean service time θ^* falls somewhere between zero and the mean interarrival time. Therefore, without any prior knowledge, the three initial guesses of θ are set to 0.25, 0.5, and 0.75 of the mean interarrival time for all simulation runs. The value of 0.7 is used for the momentum α when updating the mean service time.

For each iteration, the algorithm repeatedly uses IPA technique to estimate the derivative of the objective function. The number of customers L used by IPA are set to 40, 80, 160, and 320. Smaller values of L are not used due to the relatively large noise in IPA estimations. Higher values of L generally produce better estimates but result in longer simulation runs. To terminate the simulation, the stopping criteria with a threshold value of 0.02 is used.

Another important parameter that influences the performance of the algorithm is the window length. It has to be large enough to take advantage of all the data available and yet small enough to discard the data that are far away from the current estimate. The window size of 90 is chosen after experimenting with different sizes.

Theoretically the mean service time cannot be less than zero or greater than $\frac{1}{\lambda}$, the mean interarrival time. In practice θ will sometimes goes out of bound when the optimum solution is located near the boundary. To prevent the system from going into the meaningless ($\theta \le 0$) or unstable ($\theta \ge \frac{1}{\lambda}$) states, the following heuristics are used:

if $\theta_i \le 0$ then $\theta_i = unif(\theta_{i-1}, \frac{1}{\lambda})$

if $\theta_i \ge \frac{1}{\lambda}$ then $\theta_i = unif(0, \theta_{i-1})$

where *unif*(a,b) is uniform random variate in (a,b). The reason for this heuristic is as follow. When the estimated θ goes out of bound, the algorithm has overshoot the optimum point. A single point is randomly placed on the opposite side in an attempt to correct the problem.

4.3 Simulation Results

For each set of parameters, the simulation is run 40 times with different sets of random numbers. The results are averaged and presented in the tables below. Each entry of the tables represents the average of 40 simulations. Since the number of customers L used in QLS algorithm is not the same as those used in PARMSR algorithm in [5], comparison of the algorithm performance with respect to L cannot be made. However, the performance of both algorithms with respect to C_1 is shown graphically in Figure 1 and Figure 2. Due to the proportionality of C_1 and ρ^* as shown in Table 1, these graphs can also be interpreted as the performance of the algorithms with respect to the optimum utilization. Notice that the grand average of the percent error in J and the run length are 1.23% and 4895 customers for the QLS algorithm as compared to 1.24% and 49961 customers for PARMSR algorithm. This means that to achieve the same level of accuracy, QLS algorithm is more than 10 times faster than PARMSR algorithm.

			C_1			
L	2000	8000	32000	128000	512000	mean
40	0.33	1.22	1.92	7.52	3.89	2.98
80	0.24	0.48	0.94	0.82	1.18	0.73
160	0.13	0.39	0.47	0.22	3.87	1.02
320	0.07	0.27	0.20	0.19	0.26	0.20
mean	0.19	0.59	0.88	2.19	2.30	1.23

Table 2. Average % error in J.

			C_1			
L	2000	8000	32000	128000	512000	mean
40	0.51	2.45	1.92	10.32	0.98	3.24
80	0.38	0.67	0.95	0.85	0.47	0.66
160	0.21	0.53	0.62	0.29	18.19	3.97
320	0.10	0.57	0.25	0.43	0.35	0.34
mean	0.30	1.06	0.75	2.97	5.00	2.03

Table 3. Standard deviation of % error in J.

			C_1			
L	2000	8000	32000	128000	512000	mean
40	682	683	707	2922	5236	2046
80	1342	1608	1114	2444	10206	3343
160	2480	3088	2644	4092	15624	5586
320	5408	6720	5928	8520	16440	8603
mean	2478	3025	2598	4495	11877	4895

Table 4. Average run length (number of customers).

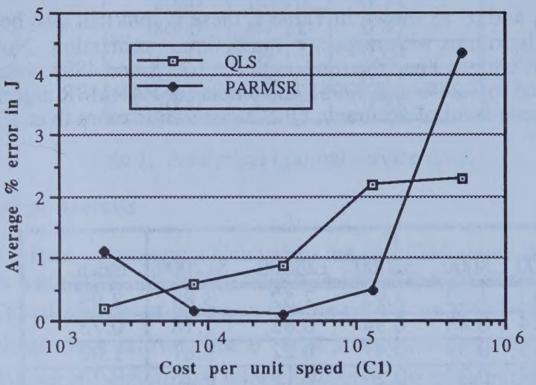


Figure 1. The average % error in J for various values of C_1.

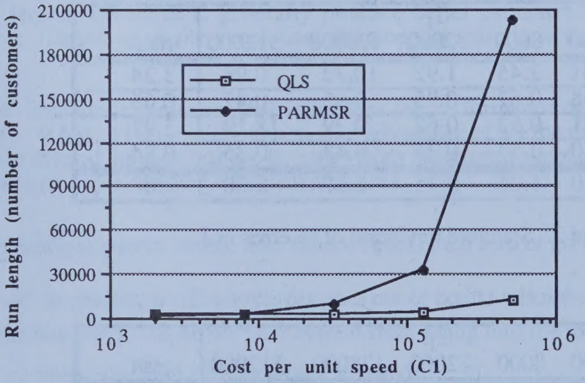


Figure 2. The average number of customers for various values of C_1.

5 Conclusion

A recursive IPA based quadratic curve fit algorithm is proposed to optimize discrete event dynamical systems. Empirical study using M/M/1 queues shows that the algorithm converges over a wide range of experiments. This algorithm is as effective as other previously known algorithms and is far more efficient for the test problem studied. More works are needed in extending the algorithm to more complex systems.

Acknowledgments

The original idea of using IPA and global response surface approach to optimization was suggested to the authors by Professor Y.C. Ho. when he taught a course at the University of Texas during the fall of 1989 ([3], Appendix C). We wish to acknowledge his original idea and many helpful suggestions during the course of experimentally implementing and investigating the approach.

References

[1] E.K.P. Chong and P.J. Ramadge, "Convergence of Recursive Optimization Algorithms Using IPA Derivative Estimates", Department of Electrical Engineering, Princeton University, 1989.

[2] M.C. Fu and Y.C. Ho, "Using Perturbation Analysis for Gradient Estimation, Averaging, and Updating in a Stochastic Approximation Algorithm for the Optimization of System Time in a G/G/1 Queue", Division of Applied Sciences, Harvard University, 1988.

[3] Y.C. Ho, and X. Cao, Perturbation Analysis of Discrete Event Dynamic Systems, Kluwer Publishing, 1990.

[4] Y.C. Ho, M.A. Eyler, and T.T. Chien, "A Gradient Technique for General Buffer Storage Design in a Serial Production Line", Int. J. of Prod. Res., v.17, no.6, pp.557-580, 1979.

[5] R. Suri and T.Y. Leung, "Single Run Optimization of Discrete Event Simulations — An Empirical Study Using the M/M/1 Queue", IIE Transactions, Vol. 21, pp.35-49, 1989.

[6] R. Suri and M.A. Zazanis, "Perturbation Analysis Gives Strongly Consistent Sensitivity Estimates for the M/G/1 Queue", Manag. Sci., v.34, no.1, pp.39-64, 1988.

[7] H.A. Taha, Operations Research: An Introduction, Ch. 13, 2nd ed., Macmillan, New York, 1976.

[8] M.A. Zazanis and R. Suri, "Estimating Second Derivatives of Performance Measures for G/G/1 Queues from a Single Sample Path", Division of Applied Sciences, Harvard University, 1985.

On A Stochastic Optimization Algorithm using IPA Which Updates After Every Customer

Edwin K. P. Chong *Peter J. Ramadge*

Department of Electrical Engineering
Princeton University, Princeton, NJ 08544

Abstract

In this paper, we resolve an issue which has remained an open problem for some time—we prove convergence (with probability one) of a stochastic optimization algorithm using IPA for an M/M/1 queue which updates after every customer. The same result can be shown to hold for a class of GI/G/1 queues. The proof can be extended to the case where updating is done after a fixed number of customers. Our method of proof also suggests that the sample path behavior of the algorithm is similar to that of an algorithm which updates after every busy period. We provide simulation results which illustrate our theorems.

1. Introduction

Sensitivity analysis of discrete event systems is an area which has received much attention in recent years. A result of such efforts has been the development of efficient methods for estimating derivatives of certain performance measures from observations of a single sample path of a discrete event system. Among these techniques are *Infinitesimal Perturbation Analysis* (IPA) [1], and the *Likelihood Ratio* (LR) method [2]. While conventional techniques for derivative estimation involve performing at least two experiments on the system, these methods require only performing one experiment, and certain problems associated with conventional techniques are alleviated. In this paper, we will be concerned only with the technique of IPA. This technique has been studied by a number of researchers (see for example [3],[4],[1],[5]).

Consider a discrete event system, and suppose that its performance is described by a performance measure $J(\theta)$ which depends on a control parameter θ. For example, the system may be a single server queue, where $J(\theta)$ is the steady state mean system time, and θ is the service rate. Suppose we wish to minimize the performance measure J. Information about the derivative $dJ/d\theta$, which can be estimated using IPA by observing a single sample path of the system, may be used to modify the control parameter θ so that it is closer to the optimal value. If the derivative estimation technique is used in conjunction with a recursive stochastic optimization algorithm (e.g. Robbins-Monro [6]), we could repeatedly estimate the derivative and update the control parameter, and the system could be optimized in a single run. The recursive algorithm would have the form $\theta_{n+1} = \theta_n - a_n \hat{h}_{n+1}$ where θ_n is the control parameter iterate, and $\hat{h}_{n+1}$ is an estimate of the derivative $h(\theta_n) = dJ(\theta_n)/d\theta$. Such a scheme could be used for on-line optimization of the system, or for single-run optimization in simulation.

Several studies have been done on using IPA derivative estimates in recursive optimization algorithms (e.g., [7], [8], [9], [10]). However, rigorous results on the convergence on such algorithms have been scarce. Fu [9] has proved convergence of an algorithm using IPA for a GI/G/1 queue, which updates after every busy period of the queue. Chong and Ramadge [10] have also provided proofs of convergence of algorithms using IPA for an M/G/1 queue, and a multi-queue system. Like Fu's algorithm, their algorithm for the M/G/1 queue also updates after every busy period. A source of concern in such an algorithm is the possibility of extremely long busy periods, especially if the load is high.

Suri and Leung [8] have performed an extensive empirical study on an algorithm using IPA for an M/M/1 queue, which updates after a fixed number of customers. They found the algorithm to converge in all the cases considered, with a relatively fast rate of convergence. If we update after every customer, then such an algorithm would always update more often than an algorithm which updates after every busy period. However, proving the convergence of such a scheme has remained an open problem for some time.

The above open problem is resolved in this paper, with a central focus on the M/M/1 queue. This paper summarizes a result in [11], which provides a proof for the M/G/1 queue. The algorithm considered is essentially the same as that of [8] (it differs only in terms of the projections, which we will discuss in the next section). Our proof can be extended to include algorithms which update after a fixed number of customers. With additional assumptions, our result extends to the GI/G/1 queue as well. In addition to proving convergence of the algorithm, the method of proof also demonstrates an interesting property of the sample path behavior of the algorithm. It suggests that the algorithm behaves in a sample path-wise similar fashion to an algorithm that updates after every busy period of the queue. This observation in turn suggests that the rates of convergence of the two algorithms are similar. We provide simulation results which illustrate this property of the algorithm.

2. Problem Formulation

2.1. Model for Controlled M/M/1 Queue

We wish to consider an M/M/1 queue, with service times that depend on a control parameter θ. For simplicity, we will assume in this paper that θ is the mean the service time. Other types of control parameters can be treated in a similar way. The probability space and corresponding sample path to model the queue can be set up in many ways. Here we will use a particular nonstandard construction which is convenient for our purposes.

We assume that the underlying probability space $(\Omega, \mathcal{F}, P)$ is equipped with the following random variables: For each $k \in \mathbb{N}$ (where $\mathbb{N} = \{1,2,\dots\}$), are two sequences $\{A_n^k\}_{n=1}^{\infty}$ and $\{U_n^k\}_{n=1}^{\infty}$. Each A_n^k is exponentially distributed, with rate λ. The A_n^k are independent over n and k. Each U_n^k is uniformly distributed in $(0,1]$. We assume that the U_n^k are independent over both n and k, and are also independent of the A_n^k.

We use the above random variables to model our M/M/1 queue as follows. We index the customers that arrive at the queue by $n = 1,2,\dots$. For convenience, we assume that the first customer arrives to find the queue empty. We will use the symbol A_n for the duration of time from the arrival of the nth customer to that of the $(n+1)$th customer, θ_n for the value of the control parameter for the nth customer, and $X_n(\theta_n)$ for the service time of the nth customer. The queue generates a sequence of alternating *busy periods* (BPs) and *idle periods* (IPs). We refer to each BP and IP which immediately follows as a *busy cycle* (BC). We index the BCs by $k = 1,2,\dots$, and the customers within each BC by $m = 1,2,\dots$. For each BC k, A_m^k is the duration between the times of arrival of the mth and $(m+1)$th customer. The service time of the mth customer in BC k is $X_m^k(\theta) = -\theta \log(U_m^k)$, when the value of the control parameter is θ. It is easily checked that $X_n^k(\theta)$ is indeed exponentially distributed with mean θ. Let $\bar{N}_k$ be the number of customers served in the kth BC. Note that the sequence of interarrival times $\{A_n\}$ can be written as

$$A_1^1, A_2^1, \dots, A_{\bar{N}_1}^1, A_1^2, A_2^2, \dots$$

while the sequence of service times $\{X_n(\theta_n)\}$ can be written as

$$X_1^1(\theta_1), X_2^1(\theta_2), \dots, X_{\bar{N}_1}^1(\theta_{\bar{N}_1}), X_1^2(\theta_{\bar{N}_1+1}), X_2^2(\theta_{\bar{N}_1+2}), \dots$$

We assume that θ_1 is a random variable independent of A_n^k and U_n^k for all n and k (it may indeed be a fixed value). The value of the control parameter θ may differ from customer to customer. Therefore, the system differs from a conventional M/M/1 queue in that the service times are not necessarily identically distributed. If the θ values for different customers were dependent random variables, then their service times would also be dependent. As we shall describe in the next section, for the problem we consider in this paper, it is indeed the case that the θ values for different customers are dependent random variables.

The number of customers served in the kth BP, $\bar{N}_k$, is different from that of a conventional M/M/1 queue, since the control parameters are varying from customer to customer. A case of particular interest is when the control parameter is held constant at a value θ throughout the entire kth BP. Then, this BP is indeed that of a conventional M/M/1 queue, since in this case the service times are independent and identically distributed. Let $N_k(\theta)$ be the number of customers served in the kth BP, assuming that the control parameter were fixed at the value θ. The expected value of $N_k(\theta)$ is given by $E(N_k(\theta)) = 1/(1-\lambda\theta)$ (see [12], p. 217).

Define a filtration (an increasing sequence of σ-algebras) $\{\mathcal{F}_n\}$ as follows[1]: $\mathcal{F}_1 = \sigma(\theta_1)$, $\mathcal{F}_2 = \sigma(\mathcal{F}_1, X_1(\theta_1))$ and for $n > 2$, $\mathcal{F}_n = \sigma(\mathcal{F}_{n-1}, A_{n-2}, X_{n-1}(\theta_{n-1}))$. So, $\mathcal{F}_n$ represents all the information about customers available at the start of the service of the nth customer (the arrival and service times of all customers up to but not including the nth customer).

2.2. Stochastic Optimization of Control Parameter

We consider the problem of selecting the control parameter θ for the above M/M/1 queue to minimize the performance measure $J(\theta)$ given by

$$J(\theta) = T(\theta) + C(\theta)$$

with $T(\theta)$ the steady-state mean system time, and $C(\theta)$ a cost function on θ. To perform the optimization, we propose to use IPA derivative estimates for $dT(\theta)/d\theta$ together with a stochastic optimization algorithm. The stochastic optimization algorithm is of the form

$$\theta_{n+1} = \theta_n - a_n \hat{h}_{n+1}$$

where, as defined before, θ_n is the control parameter for the service time of customer n, a_n is a step-size, and $\hat{h}_{n+1}$ is an $\mathcal{F}_n$-measurable estimate of $dJ/d\theta$ (based on IPA). We will discuss these quantities in detail in the next two sections. Note in particular that the algorithm above updates the control parameter after every customer.

In general, the control parameter θ_n may not assume any arbitrary value, but is constrained to lie in some set. For the example we are considering, since θ is the mean of the service time, then for stability of the queue, we must have that $0 \leq \theta < 1/\lambda$. We denote by D the constraint set in which the parameter values $\{\theta_n\}$ are required to lie. We assume D to be a compact interval of the real line $\mathbb{R}$, and we write $D = [\theta_{\min}, \theta_{\max}]$, where $\theta_{\max} > \theta_{\min} \geq 0$. The algorithm above does not ensure the invariance of D, i.e., θ_n may leave the region D. We overcome this by introducing *projections*—if during the execution of the recursive algorithm the sequence leaves D, say $\theta_{k-1} \in D$ but $\theta_k \notin D$, then we replace the obtained value of θ_k by θ_{k-1} (other forms of projections may also be possible). The algorithm is then of the form

$$\theta_{n+1} = \pi_{n+1}[\theta_n - a_n \hat{h}_{n+1}]$$

where $\pi_{n+1}[\cdot]$ is defined by

$$\pi_{n+1}[x] = \begin{cases} x, & \text{if } x \in D \\ \theta_n, & \text{otherwise} \end{cases} \tag{1}$$

---

[1] We use the notation $\sigma(Y)$ to denote the σ-algebra generated by Y, where Y may be random variable, a σ-algebra, or a combination of both.

and θ_1 is a random variable taking values in D.

Our objective is to prove convergence of the above algorithm, i.e., to prove that θ_n converges a.s. to θ^*, where θ^* minimizes the objective function $J(\theta)$. In the next three sections, we show that under certain assumptions, we can indeed prove that the algorithm converges a.s. to the optimal parameter value θ^*.

2.3. Assumptions

In this section, we state the conditions on the various components of the system that we will assume throughout this paper. The first of these assumptions concerns the step-size sequence $\{a_n\}$ used in the stochastic optimization algorithm.

Assumptions on $\{a_n\}$:

(G1) $\{a_n\}$ is adapted to $\{\mathcal{F}_n\}$ and is non-increasing;

(G2) There exists δ, $\frac{1}{2} < \delta \leq 1$, and $\bar{A} < \infty$, $\underline{A} > 0$ such that for each n, $\underline{A} n^{-\delta} \leq a_n \leq \bar{A} n^{-\delta}$ a.s.;

(G3) For all n, $\left(\frac{1}{a_{n+1}} - \frac{1}{a_n}\right) \leq B_a$ where B_a is a finite constant.

We next make an assumption to ensure that the system is stable for any parameter value $\theta \in D$.

Assumption on stability:

(S1) $\theta_{\max} < 1/\lambda$.

Finally, we make certain assumptions on the cost function C, which appears in the performance measure J.

Assumptions on $C(\theta)$:

(C1) $C(\theta)$ is differentiable with respect to θ on D;

(C2) Let $C'(\theta) = dC(\theta)/d\theta$. Then $C'(\cdot)$ is Lipschitz continuous on D, i.e., there exists $C''_m < \infty$ such that $\forall \theta_1, \theta_2 \in D$, $|C'(\theta_2) - C'(\theta_1)| \leq C''_m |\theta_2 - \theta_1|$;

(C3) Let $T'(\theta) = dT(\theta)/d\theta$, and $h(\theta) = T'(\theta) + C'(\theta)$. Then $h(\theta) = 0$ only if $\theta = \theta^*$ for some unique $\theta^* \in \overset{\circ}{D}$ (where $\overset{\circ}{D}$ denotes the interior of D), and θ^* minimizes J on D.

2.4. Stochastic Optimization Algorithm

In this section, we describe in detail the stochastic optimization algorithm using IPA derivative estimates applied to the M/M/1 queue.

We first make a few definitions. Define $S(k)$ by $S(1) = 0$, and for $k > 1$, $S(k) = \sum_{i=1}^{k-1} \bar{N}_i$. $S(k) + n$ is the index of the nth customer in the kth BP. Let $H = \{S(k) + 1 : k = 0, 1, 2, \ldots\}$ be the set of indices of customers which initiate BPs.

Let $\phi_n = X_n(\theta_n)/\theta_n$. Define $\{\hat{g}_{n+1}\}$ by

$$\hat{g}_{n+1} = \begin{cases} \phi_n, & \text{if } n \in H \\ \hat{g}_n + \phi_n, & \text{otherwise} \end{cases} \tag{2}$$

Note that for each k, and each n such that $S(k) + 1 \leq n \leq S(k) + \bar{N}_k$, $\hat{g}_{n+1} = \sum_{i=S(k)+1}^{n} \phi_i$. Readers familiar with IPA will recognize this expression as the sample derivative of the system time for customer n. Note that if the nth customer in the queue is the mth customer within BP k, then $\phi_n = -\log(U_m^k)$, and therefore ϕ_n does not depend on θ_n. The following result from [7] states how the above derivative estimate is related to the true mean system time.

Lemma 1: Suppose condition (S1) holds. Then the steady-state mean system time $T(\theta)$ is continuously differentiable on D, and the derivative $T'(\theta)$ satisfies, for every $k \in \mathbb{N}$,

$$T'(\theta) = \frac{E(\beta_k(\theta))}{E(N_k(\theta))}$$

where

$$\beta_k(\theta) = \sum_{i=1}^{N_k(\theta)} \sum_{j=1}^{i} \phi_j^k \tag{3}$$

with $\phi_j^k = X_j^k(\theta)/\theta$.

Proof: See [7]. ∎

As we have stated before, the stochastic optimization algorithm is defined by

$$\theta_{n+1} = \pi_{n+1}[\theta_n - a_n \hat{h}_{n+1}] \tag{4}$$

where $\hat{h}_{n+1}$ is an estimate of $h(\theta_n)$. The sequence $\{\hat{h}_{n+1}\}$ is defined recursively as

$$\hat{h}_{n+1} = \begin{cases} \hat{g}_{n+1} + C'(\theta_n), & \text{if } n \in H \text{ or } \theta_n = \theta_{n-1} - a_{n-1}\hat{h}_n \\ \hat{g}_{n+1} + C'(\theta_n) + \hat{h}_n, & \text{otherwise} \end{cases} \tag{5}$$

Notice that if the update after the $(n-1)$th customer is a projection, then in the nth step we include the value of $\hat{h}_n$ in our estimate $\hat{h}_{n+1}$ (unless the nth customer is the first in a BP). In this way, we use information which would otherwise have been lost in the projection. It turns out that this method of projection also helps to simplify the proof of convergence of the algorithm. In the algorithm used by Suri and Leung [8], the projection method does not use information which is lost due to projections. This aspect of the algorithm is essentially the only difference between our algorithm and the one in [8].

3. Convergence of Stochastic Optimization Algorithm

3.1. Main Result

With the assumptions given in the foregoing sections, we now state the main result.

Theorem 1: Suppose that (G1–G3), (S1), and (C1–C3) are satisfied. Then, for the sequence $\{\theta_n\}$ defined by (2), (4), and (5), we have that $\theta_n \to \theta^*$ as $n \to \infty$ with probability one.

3.2. Idea of Proof of Main Result

To prove Theorem 1, we need the following definitions. Let $\tilde{\theta}_n = \theta_{S(n)+1}$, $\tilde{a}_n = a_{S(n)+1}$, and $\tilde{\pi}_n[\cdot] = \pi_{S(n)+1}[\cdot]$. The sequences $\{\tilde{\theta}_n\}$, $\{\tilde{\pi}_n\}$ and $\{\tilde{a}_n\}$ are subsequences of $\{\theta_n\}$, $\{\pi_n\}$ and $\{a_n\}$, respectively, taken at the start of every BP. Let $X^n = \{U_m^n : m \in \mathbb{N}\}$ and $A^n = \{A_m^n : m \in \mathbb{N}\}$. Define a filtration $\{\tilde{\mathcal{F}}_n\}$ as follows: $\tilde{\mathcal{F}}_1 = \sigma(\theta_1)$ and for $n > 1$, $\tilde{\mathcal{F}}_n = \sigma(\tilde{\mathcal{F}}_{n-1}, X^{n-1}, A^{n-1})$.

The proof of Theorem 1 involves the following steps:

1. Showing that $\{\tilde{\theta}_n\}$ satisfies the recursion

   $$\tilde{\theta}_{n+1} = \tilde{\pi}_{n+1}[\tilde{\theta}_n - \tilde{b}_n \tilde{h}_{n+1}]$$

   where $\{\tilde{b}_n\}$ is not summable a.s., but is square summable, and $\tilde{h}_{n+1} = h(\tilde{\theta}_n) + \tilde{\varepsilon}_{n+1}$ where $\{\tilde{\varepsilon}_{n+1}\}$ is "small" (in a sense to be defined later);
2. Showing that $\tilde{\theta}_n \to \theta^*$ a.s.; and

3. Showing that the whole sequence $\{\theta_n\}$ converges to θ^* a.s.

Due to space limitations, it is not possible to include the proofs of the above steps. We shall only highlight the main ideas involved in the proofs. For detailed proofs, see [11]. We first consider Step (1) by stating it formally below, this being the most crucial step.

Lemma 2: Suppose the assumptions of Theorem 1 are satisfied. Then, there exist sequences $\{\tilde{h}_n\}$ and $\{\tilde{b}_n\}$ of random variables adapted to $\{\tilde{\mathcal{F}}_n\}$ such that

(a) $\{\tilde{\theta}_n\}$ satisfies the recursion

$$\tilde{\theta}_{n+1} = \tilde{\pi}_{n+1}[\tilde{\theta}_n - \tilde{b}_n\tilde{h}_{n+1}] \tag{6}$$

where

(b) $\sum_{n=1}^{\infty}\tilde{b}_n = \infty$ and $\sum_{n=1}^{\infty}\tilde{b}_n^2 < \infty$;

If we let $\tilde{\varepsilon}_{n+1} = \tilde{h}_{n+1} - h(\tilde{\theta}_n)$, then

(c) $\sum_{n=1}^{\infty}\tilde{b}_n|E_{\tilde{\mathcal{F}}_n}(\tilde{\varepsilon}_{n+1})| < \infty$ a.s., and

(d) $E_{\tilde{\mathcal{F}}_n}(\tilde{\varepsilon}_{n+1}^2) \leq \sigma^2 < \infty$.

Idea of Proof: For simplicity, we shall ignore the projections. So, $\theta_{n+1} = \theta_n - a_n\hat{h}_{n+1}$ where $\hat{h}_{n+1} = \hat{g}_{n+1} + C'(\theta_n)$. Projections can be included in the analysis, but with additional notation and definitions.

Consider the first BP, i.e., fix $n = 1$ (this simplifies the notation, although the following argument applies to every BP). Note that $S(1) = 0$, $\tilde{\theta}_1 = \theta_1$, and $\tilde{a}_1 = a_1$. To further simplify the notation, let $N = N_1(\theta_1)$, $\bar{N} = \bar{N}_1$, and $N_{\max} = N_1(\theta_{\max})$.

By using the definitions of $\tilde{\theta}_2$ and $\tilde{\pi}_2$, we get

$$\begin{aligned}
\tilde{\theta}_2 &= \theta_{\bar{N}+1} \\
&= \pi_{\bar{N}+1}[\theta_{\bar{N}} - a_{\bar{N}}\hat{h}_{\bar{N}+1}] \\
&= \tilde{\pi}_2[\theta_{\bar{N}} - a_{\bar{N}}\hat{h}_{\bar{N}+1}] \\
&= \tilde{\pi}_2\left[\theta_1 - \sum_{i=1}^{\bar{N}} a_i(\hat{g}_{i+1} + C'(\theta_i))\right]
\end{aligned}$$

Now,

$$\begin{aligned}
\sum_{i=1}^{\bar{N}} a_i(\hat{g}_{i+1} + C'(\theta_i)) &= \sum_{i=1}^{\bar{N}} a_i\left(\sum_{j=1}^{i}\phi_j + C'(\theta_1)\right) \\
&= a_1E_{\theta_1}(N)\sum_{i=1}^{\bar{N}}\frac{a_i}{a_1}\left(\sum_{j=1}^{i}\phi_j + C'(\theta_i)\right)\frac{1}{E_{\theta_1}(N)} \\
&= \tilde{a}_1E_{\theta_1}(N)\left(\frac{1}{E_{\theta_1}(N)}\sum_{i=1}^{\bar{N}}\frac{a_i}{a_1}\sum_{j=1}^{i}\phi_j + \frac{1}{E_{\theta_1}(N)}\sum_{i=1}^{\bar{N}}\frac{a_i}{a_1}C'(\theta_i)\right) \\
&= \tilde{a}_1E_{\theta_1}(N)(\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3)
\end{aligned}$$

where $\tilde{c}_1$, $\tilde{c}_2$ and $\tilde{c}_3$ are defined as follows:

$$\tilde{c}_1 = \frac{1}{E_{\theta_1}(N)}\sum_{i=1}^{N}\frac{a_i}{a_1}\sum_{j=1}^{i}\phi_j$$

$$\tilde{c}_2 = \frac{1}{E_{\theta_1}(N)} \sum_{i=1}^{N} \frac{a_i}{a_1} C'(\theta_i)$$

$$\tilde{c}_3 = \begin{cases} \frac{1}{E_{\theta_1}(N)} \sum_{i=N+1}^{\bar{N}} \frac{a_i}{a_1} \left(\sum_{j=1}^{i} \phi_j + C'(\theta_i) \right), & \text{if } \bar{N} > N \\ 0, & \text{if } \bar{N} = N \\ -\frac{1}{E_{\theta_1}(N)} \sum_{i=\bar{N}+1}^{N} \frac{a_i}{a_1} \left(\sum_{j=1}^{i} \phi_j + C'(\theta_i) \right), & \text{if } \bar{N} < N \end{cases}$$

Let $\tilde{h}_2 = \tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3$, and $\tilde{b}_1 = E_{\tilde{\theta}_1}(N)\tilde{a}_1$. Then, we may write $\tilde{\theta}_2 = \tilde{\pi}_2[\tilde{\theta}_1 - \tilde{b}_1 \tilde{h}_2]$. It is easy to check that $\tilde{h}_2$ is $\tilde{\mathcal{F}}_2$-measurable. This proves (a).

To prove (b), we first note that by (S1), we can write $1 \le E_{\tilde{\theta}_1}(N) \le E(N_{\max}) < \infty$. Then, by using property (G2) of $\{a_n\}$, we can show that $\{\tilde{b}_n\}$ has the required properties in (b).

To show (c), we first write $\tilde{\varepsilon}_2$ as

$$\tilde{\varepsilon}_2 = (\tilde{c}_1 - T'(\theta_1)) + (\tilde{c}_2 - C'(\theta_1)) + \tilde{c}_3$$

It will suffice to show that $|E_{\mathcal{F}_1}(\tilde{c}_1 - T'(\theta_1))| \le B_1 a_1$, $|E_{\mathcal{F}_1}(\tilde{c}_2 - C'(\theta_1))| \le B_2 a_1$ and $|E_{\mathcal{F}_1}(\tilde{c}_3) \le B_3 a_1^d$, where $d = 1/(2\delta)$, since we may then write

$$\sum_{n=1}^{\infty} \tilde{b}_n |E_{\tilde{\mathcal{F}}_n}(\tilde{\varepsilon}_{n+1})| \le \sum_{n=1}^{\infty} E(N_{\max}) \left((B_1 + B_2) a_n^2 + B_3 a_n^{(1+d)} \right) < \infty$$

Now, using Lemma 1,

$$\begin{aligned} E_{\mathcal{F}_1}(\tilde{c}_1 - T'(\theta_1)) &= \frac{1}{E_{\theta_1}(N)} E_{\mathcal{F}_1} \left(\sum_{i=1}^{N} \frac{a_i}{a_1} \sum_{j=1}^{i} \phi_j \right) - \frac{1}{E_{\theta_1}(N)} E_{\mathcal{F}_1} \left(\sum_{i=1}^{N} \sum_{j=1}^{i} \phi_j \right) \\ &= \frac{a_1}{E_{\theta_1}(N)} E_{\mathcal{F}_1} \left(\sum_{i=1}^{N} \left(\frac{1}{a_1} - \frac{1}{a_i} \right) \sum_{j=1}^{i} \phi_j \right) \end{aligned}$$

and therefore,

$$|E_{\mathcal{F}_1}(\tilde{c}_1 - T'(\theta_1))| \le a_1 E_{\mathcal{F}_1} \left(\sum_{i=1}^{N} \left| \frac{1}{a_1} - \frac{1}{a_i} \right| \sum_{j=1}^{i} \phi_j \right)$$

It is easy to show using (G3) that

$$\left| \frac{1}{a_1} - \frac{1}{a_i} \right| \le N_{\max} B_a$$

and hence

$$|E_{\mathcal{F}_1}(\tilde{c}_1 - T'(\theta_1))| \le a_1 B_1$$

where

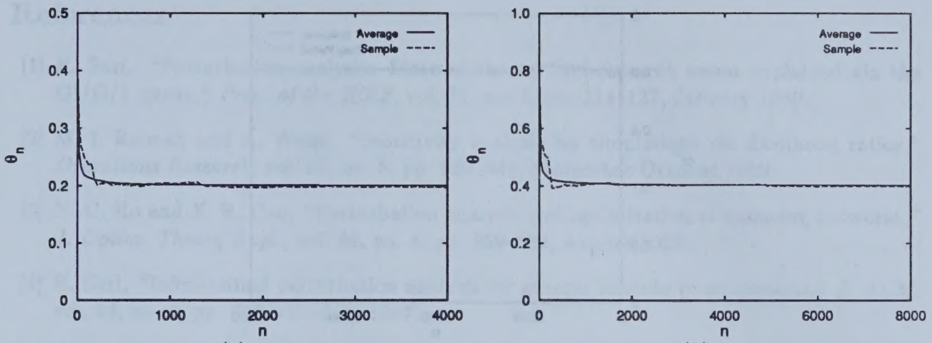
$$B_1 = B_a E \left(N_{\max}^2 \sum_{i=1}^{N_{\max}} \phi_j \right) < \infty$$

With a similar argument, and using (C2), it is possible to show that

$$|E_{\mathcal{F}_1}(\tilde{c}_2 - C'(\theta_1))| \le a_1 B_2$$

It remains to show that $|E_{\mathcal{F}_1}(\tilde{c}_3)| \le B_3 a_1^d$, i.e., as n gets large, $|E_{\mathcal{F}_n}(\tilde{c}_3)|$ becomes small. To see this, we only need to note that for large n, the θ_n do not change by much (since a_n becomes small), and $\bar{N}$ will be equal to N with high probability.

It is straightforward to show (d), simply by using the fact that $N_{\max}$ and $\sum_{i=1}^{N_{\max}} \phi_i$ have finite moments. We omit the details here. ∎



(a) (b)

Figure 1: θ_n versus n for the M/M/1 Queue ((a) $\theta^* = 0.2$, (b) $\theta^* = 0.4$)

We next state Step (2) formally in the following lemma.

Lemma 3: Suppose the assumptions of Theorem 1 hold. Then, $\tilde{\theta}_n \to \theta^*$ as $n \to \infty$ with probability one.

Idea of Proof: From the result of Step (1), Step (2) follows immediately by applying a convergence result from stochastic approximations. A detailed proof for such a result can be found in [13]. ∎

Step (3) follows from Step (2) by showing that the values of θ_n within each BP are "close" to the values at the beginning of the BP (the $\tilde{\theta}_n$). Specifically, for each n let K_n be the index of the first customer in the BP containing the nth customer. Then for each $\varepsilon > 0$ we can show that the sequence of probabilities $\{P\{|\theta_n - \theta_{K_n}| \geq \epsilon\}\}$ is summable, and hence by the Borel-Cantelli Lemma, $|\theta_n - \theta_{K_n}| \to 0$ a.s. Since by Lemma 3 $|\theta_{K_n} - \theta^*| \to 0$ a.s., then we conclude that $\theta_n \to \theta^*$ a.s., which completes the proof of main theorem.

Step (1) basically shows that the sequence $\{\tilde{\theta}_n\}$ behaves like an algorithm which updates after every BP, and Step (2) is essentially a convergence result for such an algorithm. This suggests that the rate of convergence of the algorithm is essentially that of one which updates after every BP. In the next section, we will see this property illustrated in simulation results.

4. Simulations

In this section, we provide simulation results which illustrate the main theorem of this paper.

Let D be chosen such that (S1) is satisfied. Consider the problem of optimizing $J(\theta) = T(\theta) + C(\theta)$ where $C(\theta) = c/\theta$. Note that assumptions (C1–C3) are satisfied for this cost function C. For the stochastic optimization algorithm, we use a sequence $\{a_n\}$ given by $a_n = a/n$ where $a \in \mathbb{R}$ is some constant. Assumptions (G1–G3) hold for this sequence. Therefore, all the assumptions of Theorem 1 hold, and convergence of the algorithm is guaranteed.

Figure 1(a) shows θ_n versus n for the algorithm applied to the M/M/1 queue. The solid curve represents an average over 100 sample paths of the algorithm, whereas the dashed curve is a single sample path. The following values were used: $a = 0.05$, $c = 0.0625$, $\lambda = 1$, $D = [0.0031, 0.952]$, $\theta_1 = 0.4$. For the case of the M/M/1 queue, we can evaluate the optimal value θ^* analytically to be $\theta^* = 1/(\lambda + 1/\sqrt{c})$. For the values used above, $\theta^* = 0.2$. Figure 1(b) shows θ_n versus n with the same values as above, except $c = 0.4444$ and $a = 0.03$. In this case, $\theta^* = 0.4$. Experimental convergence is clearly observed in both these figures.

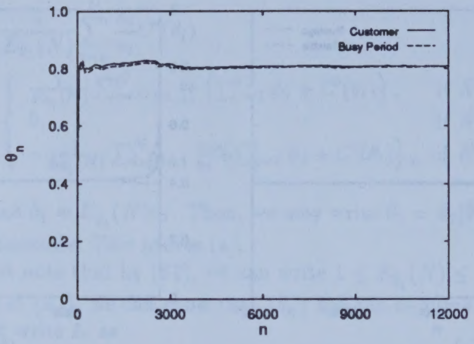


Figure 2: A Comparison of the "Customer" and "Busy Period" algorithms

To compare the convergence rate of the algorithm (which updates after every customer) with one which updates after every BP, we consider the following algorithm:

$$\theta_{n+1} = \pi_{n+1}[\theta_n - \tilde{b}_n \tilde{h}_{n+1}]$$

where θ_n is the control parameter value used in the nth BP, and $\{\tilde{b}_n\}$ is given by $\tilde{b}_n = \tilde{a}_n/(1 - \lambda\theta_n)$ where $\{\tilde{a}_n\}$ is the subsequence of $\{a_n\}$ taken at the start of every BP. $\tilde{h}_{n+1}$ is given by

$$\tilde{h}_{n+1} = (1 - \lambda\theta_n) \sum_{i=1}^{N_n} \sum_{j=1}^{i} \frac{X_j^n(\theta_n)}{\theta_n} + C'(\theta_n)$$

where N_n is the number of customers served in the nth BP (where all customers have parameter value θ_n), and the $X_j^n(\theta_n)$ are the service time of customers in the nth BP. The algorithm above arises naturally in the proof of Theorem 1. We will refer to the original algorithm as the "Customer" algorithm, and the one which updates after every BP as the "Busy Period" algorithm.

At low loads the length of BPs would be small most of the time, and thus the "Customer" and "Busy Period" algorithms would naturally behave similarly. Therefore, it is more interesting to compare the two algorithms when the load is high. In Figure 2 we compare the rates of convergence of the "Customer" algorithm with the "Busy Period" algorithm for a relatively high load ($\theta^* = 0.8$). The index n in the plot represents the number of customers. The parameter values used were: $a = 0.002$, $c = 16$, $\lambda = 1$, $D = [0.001, 0.999]$, $\theta_1 = 0.4$. The curves indicate that the sample paths of the two algorithms remain close to each other, and hence their rates of convergence are similar.

5. Conclusion

We have proved convergence of a stochastic optimization algorithm using IPA for an M/M/1 queue which updates after every customer. The method of proof can be extended in a straightforward (but somewhat complicated) way to prove convergence of a similar algorithm which updates after every fixed number of customers. With additional assumptions, we can also prove convergence of the algorithm applied to a GI/G/1 queue. With even more general assumptions, the method of proof may also be used to prove convergence of other classes of algorithms which update after every customer, such as those which use likelihood ratio derivative estimates, or algorithms applied to networks of queues. This area of study is currently being pursued by the authors.

References

[1] R. Suri, "Perturbation analysis: State of the art and research issues explained via the GI/G/1 queue," *Proc. of the IEEE*, vol. 77, no. 1, pp. 114–137, January 1989.

[2] M. I. Reiman and A. Weiss, "Sensitivity analysis for simulations via likelihood ratios," *Operations Research*, vol. 37, no. 5, pp. 830–844, September-October 1989.

[3] Y.-C. Ho and X. R. Cao, "Perturbation analysis and optimization of queueing networks," *J. Optim. Theory Appl.*, vol. 40, no. 4, pp. 559–582, August 1983.

[4] R. Suri, "Infinitesimal perturbation analysis for general discrete event systems," *J. ACM*, vol. 34, no. 3, pp. 686–717, July 1987.

[5] P. Glasserman, "Structural conditions for perturbation analysis derivative estimation: Finite-time performance indices," June 1989, submitted for publication.

[6] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, pp. 400–407, 1951.

[7] R. Suri and M. A. Zazanis, "Perturbation analysis gives strongly consistent sensitivity estimates for the M/G/1 queue," *Management Sci.*, vol. 34, no. 1, pp. 39–64, January 1988.

[8] R. Suri and Y. T. Leung, "Single run optimization of discrete event simulations—An empirical study using the M/M/1 queue," *IIE Transactions*, vol. 21, no. 1, pp. 35–49, March 1989.

[9] M. C. Fu, "Convergence of a stochastic approximation algorithm for the GI/G/1 queue using infinitesimal perturbation analysis," April 1989, submitted for publication.

[10] E. K. P. Chong and P. J. Ramadge, "Convergence of recursive optimization algorithms using IPA derivative estimates," in *Proc. 1990 American Control Conference*, pp. 1022–1027, May 1990.

[11] E. K. P. Chong and P. J. Ramadge, "Convergence of stochastic optimization algorithms using IPA which update after every customer," June 1990, Tech. Report, Dept. of Elec. Engr., Princeton University.

[12] L. Kleinrock, *Queueing Systems, Volume I: Theory*, New York: J. Wiley, 1975.

[13] E. K. P. Chong, "Convergence of stochastic approximations via martingales," March 1990, Chapter 2, Ph.D. Thesis (in preparation), Dept. of Electrical Engineering, Princeton University.

Extensions of perturbation analysis to stationary queues

P. Konstantopoulos
INRIA, Sophia Antipolis
2004 Route des Lucioles
06565 Valbonne Cedex
FRANCE

M. Zazanis
Dept. of Industrial Engineering
and Management Sciences
Northwestern University
Evanston, IL 60208

Abstract

Unbiased perturbation analysis estimators for general performance criteria are given for a class of queueing systems in a stationary ergodic context. The contribution of this paper over previous work in this area consists in providing simplified proofs and relaxing previous restrictive stochastic assumptions. The latter is particularly important when applying these results to general networks.

1 Introduction

Consider a queue with stationary and ergodic arrival $\{t_n\}$ and service $\{\sigma_n\}$ processes. Let θ be a parameter (in a sense that will be made precise later) of the service process such that the queue under consideration is stable for all allowable values of θ. There is considerable interest in providing reasonable estimators for derivatives of performance measures $J(\theta)$, like the variance of the workload in steady-state, with respect to the parameter θ. The term perturbation analysis estimator of $\frac{d}{d\theta}J(\theta)$ refers to the construction of a process $Y_t(\theta)$ on the same probability space as our original data $\{t_n\}$ and $\{\sigma_n\}$, from which we can, for instance, derive an estimator of the form

$$\frac{1}{t}\int_0^t Y_t(\theta)dt \tag{1}$$

that will converge, as $t \to \infty$, to $\frac{d}{d\theta}J(\theta)$.

In a renewal context, i.e., when interrarival times are i.i.d. and independent of the service times , also i.i.d. with common distribution G_θ, one starts by constructing the service time processes on the same probability space by doing the obvious thing; namely, by choosing i.i.d. numbers ξ_n, uniformly distributed on $[0,1]$ and by letting $\sigma_n(\theta) = G_\theta^{-1}(\xi_n)$, where G_θ^{-1} is a version of the inverse of the function G_θ. This way, one has constructed all processes associated with the evolution of the queue on the same probability space. Consider, for instance, the resulting workload process, denoted by $W_t(\theta)$. It has been shown (see [2] and [3]) that a perturbation analysis estimator for the derivative of the expected workload in steady-state is given by (1) where $Y_t(\theta)$ is the pathwise right derivative of $W_t(\theta)$ with respect to θ. The main difficulty in the proofs here is to show that a small change δ in the service time results in an overlap of the busy periods of the workload process of order $o(\delta)$.

In this paper we device an argument that, on one hand, overcomes this difficulty and on the other it allows us to extend the results to stationary and ergodic queues and to consider more general performance measures, like expectations of not necessarily continuous functions of the workload. Some assumptions on the parametrization of the service times are needed. These are given in the sequel.

Consider two jointly stationary and ergodic sequences $\{\tau_n\}$ and $\{\xi_n\}$ under a probability measure that will be denoted by P^0 in order to conform with the notation used in the paper. The first one is the sequence of interarrival times while the latter is the sequence of random variables from which the service times are generated in the following way: There is a deterministic function $(\xi, \theta) \to h(\xi, \theta)$ such that $\sigma_n(\theta) = h(\xi_n, \theta)$. The parameter θ ranges over a finite interval $[a, b]$ and $E^0\sigma_n(\theta) < E^0\tau_n$ for all θ. Moreover:
(i) $\theta \to h(\xi, \theta)$ is differentiable and Lipschitz, i.e., $|h(\xi, \theta_1) - h(\xi, \theta_2)| \leq K(\xi)|\theta_1 - \theta_2|$.
(ii) $h(\xi, \theta) \leq h(\xi, b)$.
(iii) $\xi \to h(\xi, \theta)$ is one-to-one.
Under these conditions (which are not necessarily the minimal ones) plus some extra moment conditions we construct estimators for the derivatives of functions of the workload and prove their strong consistency in Section 2. In Section 3 we discuss some simulation applications and we conclude by commenting on the construction of estimators of other quantities.

2 Derivative estimates for functions of the workload process in steady state

Let $\tilde{W}(\theta)$ be a random variable distributed like the steady-state workload when the parameter of the service process has the value θ. Let f be a function of bounded variation such that $Ef(\tilde{W}(\theta)) < \infty$. The questions that we want to deal with are:

- Does the derivative of $Ef(\tilde{W}(\theta))$ exist?
- If yes, is there a perturbation analysis estimator for this derivative?

Let P be the distribution of the stationary version of the marked point process $[t_n, \xi_n]$ (the n-th point t_n is the n-th arrival time and has ξ_n as a mark). Note that P^0 is in fact the Palm transformation of P with respect to the arrival process. Since $E^0\sigma_n(\theta) < E^0\tau_n$, there is a stationary workload process $W_t(\theta)$ under the probability measure P. Observe that knowlegde of $W_t(\theta)$ for some value of θ implies knowledge of it for *all* values of θ. This is due to assumption **(iii)** of Section 1 that says that if we know $\sigma_n(\theta)$ for some θ then we also know it for all θ. We can thus construct the right derivative of $W_t(\theta)$ with respect to θ pathwise. Denote this process by $W'_t(\theta)$. It is not difficult to see that this process is right continuous, piecewise constant and

$$W'_{t_n}(\theta) = (W'_{t_n-}(\theta) + \sigma'_n(\theta))1(W_{t_n-}(\theta) > 0). \tag{2}$$

Thus it jumps by an amount equal to $\sigma'_n(\theta)$ at each arrival t_n and is set to zero at those arrivals that find the system empty.

Our first sensitivity result is an expression for the derivative of the stationary probability that the workload exceeds a level x. We need some extra notation: We will denote by P_x the Palm transformation of P with respect to the point process of the downcrossings of $W_t(\theta)$ at level x. Informally speaking, this is P conditional on having an x-downcrossing at the origin of time. Note that P_x depends on θ. Let also λ_x be the rate of the x-downcrossings by $W_t(\theta)$.

In the analysis that follows it is important to bear in mind that $W_t(\theta)$ has right continuous paths. We will also need some notation regarding the process $W_t(b)$. We denote by $\{T_n(b)\}$ the times of arrivals of customers into an empty system and by P^* the Palm transformation of P with respect to those times. Finally let $A(I)$ denote the number of arrivals in a set I.

Theorem 1 *Suppose that* $E^0 A[T_0(b), T_1(b))^2 < \infty$ *and* $E^0 K(\xi_0)^2 < \infty$. *Then*

$$\frac{\partial}{\partial \theta} P(W_0(\theta) > x) = \lambda_x E_x W'_0(\theta). \tag{3}$$

Proof: Using the Palm inversion formula we get

$$P(W_0(\theta) > x) = \lambda^* E^* \int_{T_0(b)}^{T_1(b)} 1(W_t(\theta) > x)dt, \tag{4}$$

where λ^* is the rate of the point process $\{T_n(b)\}$. The reason that we use this particular point process in our argument is the essential element of the proof. The basic observation here is that $W_t(\theta) \leq W_t(b)$ for all $\theta \in [a, b]$ (due to the assumption that $\sigma_n(b)$ dominates $\sigma_n(\theta)$ for all θ) and hence the customers that find the b-system empty also find the θ-system empty, for all $\theta \in [a, b]$. Thus the dependence on θ on the right hand side of (4) is only inside the integrand and not on the integration limits. We now have

$$\frac{1}{\delta}\{P(W_0(\theta+\delta) > x) - P(W_0(\theta) > x)\} = \lambda^* E^* \int_{T_0(b)}^{T_1(b)} \frac{1}{\delta}\{1(W_t(\theta+\delta) > x) - 1(W_t(\theta) > x)\}dt. \tag{5}$$

Observe that the quantity inside the the expectation of (5) converges, pathwise as $\delta \to \infty$, to

$$\sum_{T_0(b) \leq d_n < T_1(b)} W'_{d_n}(\theta),$$

where d_n is the sequence of downcrossings of the level x by the process $W_t(\theta)$ (and hence depends on θ). This is a direct consequence of the definition of $W'_t(\theta)$ (see formula (2)). It can be easily seen that the quantity inside the expectation of (5) is bounded by

$$\frac{1}{\delta} \sum_{T_0(b) \le t_n < T_1(b)} \sum_{i=0}^{n} |\sigma_i(\theta + \delta) - \sigma_i(\theta)| := Z(\delta).$$

To show existence of the derivative of $P(W_0(\theta) > x)$ it now suffices to show that $Z(\delta)$ is bounded above by an E^*-integrable random variable (dominated convergence theorem). To this end, use the Lipschitz property to get

$$Z(\delta) \le \sum_{T_0(b) \le t_n < T_1(b)} \sum_{i=0}^{n} K(\xi_i).$$

This is further bounded above by

$$N \sum_{i=0}^{N-1} K(\xi_i),$$

where $N = A[T_0(b), T_1(b))$. Using the cycle formula (see [1]) between P^* and P^0 we find that the E^*-expectation of the latter quantity is equal to

$$\frac{\lambda}{\lambda^*} E^0 N K(\xi_0).$$

A straightforward application of the Cauchy-Schwarz inequality shows that this is dominated by

$$\frac{\lambda}{\lambda^*} \sqrt{E^0 N^2} \sqrt{E^0 K(\xi_0)^2},$$

which is finite by the assumptions. We conclude that $\frac{\partial}{\partial\theta} P(W_0(\theta) > x)$ exists and is equal to

$$\lambda^* E^* \sum_{T_0(b) \le d_n < T_1(b)} W'_{d_n}(\theta).$$

Using cycle formula once more we find that the quantity above is equal to $\lambda_x E_x W'_0(\theta)$ and thus conclude the proof of Theorem 1. □

We now extend the result of Theorem 1 to more general functions f. That is, we want to obtain an expression for $\frac{\partial}{\partial\theta} Ef(W_0(\theta))$ analogous to that of (3).

Theorem 2 *Consider a function f that is locally of bounded variation. Then under the assumptions of Theorem 1 plus additional technical assumptions, we have*

$$\frac{\partial}{\partial\theta} Ef(W_0(\theta)) = \int_0^\infty \lambda_x E_x W'_0(\theta) f(dx). \tag{6}$$

Sketch of proof: Suppose that f is an elementary function, that is, a finite linear combination of step functions. Then the (6) is a direct consequence of Theorem 1. More generally, we can approximate the function f by a suitable sequence f_n of elementary functions. Under appropriate conditions, $\frac{\partial}{\partial\theta} \lim_{n\to\infty} Ef_n(W_0(\theta)) = \lim_{n\to\infty} \frac{\partial}{\partial\theta} Ef_n(W_0(\theta))$. But the latter limit is equal to $\lim_{\to\infty} \int_0^\infty \lambda_x E_x W'_0(\theta) f_n(dx)$, which gives the right hand side of (6).

3 Extensions and remarks

We now consider briefly the queue length process $Q_t(\theta)$. In contrast to the workload process, the pointwise derivative of $Q_t(\theta)$ with respect to θ is equal to 0 for almost all t. Let $D(\theta)$ be the departure process, $S_n(\theta)$ the n'th departure epoch, and $Y_t(\theta)$ the derivative of the last departure time before t with respect to θ. Dropping the dependence on θ for simplicity, Y_t is constant in $[S_i, S_{i+1})$, and given by the following recursion on departure epochs:

$$Y_{S_{n+1}} = \sigma'_{n+1} + 1(Q_{S_n} > 0)Y_{S_n}.$$

As usual, these processes are defined to be right continuous. Finally N_k is the point process of the k-downcrossings of $Q_t(\theta)$ (downcrossings from k to $k-1$, i.e., departures that leave the system with $k-1$ customers), λ_k is the intensity of N_k, and E_k is the expectation with respect to the Palm probability of k-downcrossings. Then an analysis similar in every respect to the one in the proof of Theorem 1 gives

$$\frac{\partial}{\partial\theta}P(Q_0 \le k) = \lambda_k E_k[Y_0]. \tag{7}$$

To obtain the counterpart of (6) consider a positive increasing $f : N \to R$. Write f as $\sum_{i=1}^{\infty} \alpha_i 1(k \ge i)$ and approximate it by below by $f_n(k) = 1(k \le n)f(k)$. Then, by (7) and the linearity of expectation,

$$\frac{\partial}{\partial\theta}E[f_n(Q_0)] = \sum_{i=1}^{n} \alpha_i \lambda_i E_i[Y_0].$$

A uniform convergence argument allows us to pass in the limit:

$$\frac{\partial}{\partial\theta}E[f(Q_0)] = \sum_{i=1}^{\infty} \alpha_i \lambda_i E_i[Y_0]. \tag{8}$$

Let E^1 denote the expectation with respect to the Palm probability corresponding to departures. We now use the cycle formula in order to get another expression for (8):

$$\lambda_i E_i[Y_0] = \lambda E^1 \int_{[S_0,S_1)} Y_s N_i(ds) = \lambda E^1 Y_{S_0} 1(Q_{S_0} = k-1),$$

where we have used the fact that Y_s is constant on $[S_0, S_1)$ and that there is a k-downcrossing at S_0 if and only if $Q_{S_0} = k-1$. Writing $1(Q_{S_0} = k-1) = 1(Q_{S_0} \ge k-1) - 1(Q_{S_0} \ge k)$ we obtain

$$\frac{\partial}{\partial\theta}Ef(Q_0) = \lambda E^1 Y_{S_0}[f(Q_{S_0}-) - f(Q_{S_0})].$$

In the above expression $Q_{S_0}- = Q_{S_0} + 1$ is the number of customers in the system just before a departure.

From a simulation point of view, the results in this paper are important since they justify isolating a single server node in and using the estimators developed above. Consider a network in stationary regime and a single server node which does not belong to any feedback loop. The proposed estimators would be strongly consistent for performance criteria of the type described here.

References

[1] J. Neveu (1976). Sur les mesures de Palm de deux processus ponctuels stationnaires. *Zeit. f. W.*, **34**, 199-203.

[2] R.Suri and M. Zazanis (1988). Perturbation analysis gives strongly consistent estimates for the M/G/1 queue. *Mgmt. Sci.*, **1**, 39-64.

[3] M.Zazanis and R. Suri (1989) Perturbation analysis of the GI/G/1 queue. *QUESTA*, submitted.

PRIORITY LOAD SHARING: AN APPROACH USING STACKELBERG GAMES

A. A. ECONOMIDES and J. A. SILVESTER

Electrical Engineering - Systems Department
University of Southern California
Los Angeles, CA 90089-0781

Abstract

In this paper we consider the load sharing problem in a multiprocessor, where different classes of jobs have different priorities and each priority class optimizes its own objective function. First, we formulate the problem as a leader-follower Stackelberg game, where the high priority jobs constitute the leader and the low priority jobs constitute the follower. Then we focus on a special case of two preemptive resume priority classes that share a two-processor system. In this case, we find the Stackelberg equilibrium solution that minimizes the average job delay of each class. An interesting result is that if both classes have equal mean service requirements, when both processors are used, then the low priority load sharing decisions are independent of the arrival rates. Also, for equal mean service requirements and constant total arrival rate, the overall average delay of jobs from both classes is constant, i.e. does not depend on the mix of high and low priority jobs. Finally, we comment on our approach for multiobjective optimization of distributed systems with multi-priority classes.

1. INTRODUCTION

The usual approach to distributed system design and control is the optimization of a single function. If multiple objectives are desired, then the usual approach is to combine the objectives as seen by the system administrator [2] into a single function. Thus, it is assumed that all customers in the system are treated similarly and they cooperate for the socially optimum, such as optimizing the average customer performance. However, in a real distributed environment there is a diversity of customer classes, each with possibly different objectives and different service and accounting requirements. In [6, 7], we have taken a game theoretic approach for performance optimization of competing classes in a distributed computing system. In those papers, we have formulated and solved the routing problem among competing classes of jobs as a Nash game.

It is quite common to require differentiated service among different classes by assigning different priorities to different classes, for example interactive jobs have higher priority than batch jobs. A high priority class may acquire most of the resources that it needs, while a low priority class should wait for the high priority class to complete service. Since the reason for having priorities is to give preferential treatment to the high priority jobs, it is *not* meaningful to define a *single* multi-objective function (e.g. a convex combination of the objective functions of the different priority classes) for global optimization *across all the priority classes simultaneously.* However, we can still optimize the behavior of jobs within each priority class. Therefore a different approach should be taken for performance optimization of multipriority systems. In this paper, we formulate and optimize the performance of different priority classes as a Stackelberg game. In [4], we have considered the joint load sharing, routing and congestion control problem.

For simplicity of presentation, we consider two priority classes of customers which select between two servers. Jobs from the high priority class and jobs from the low priority class arrive to a two-processor system requiring execution. The problem of deciding to which processor each job will be assigned is the load sharing problem [9, 3] (Fig. 1).

2. QUEUEING MODEL

In this section, we introduce a simple queueing model of two servers that are shared by customers of two priority classes (Fig. 1). The problem is to assign these customers to the two servers so as to minimize the average delay of each class. An application is load sharing for a multiprocessor system, where interactive jobs (high priority) and batch jobs (low priority) may use two processors for execution. Another application is routing, where voice packets (high priority) and data packets (low priority) may use two different links for transmission between source-destination.

Let the high priority class α jobs arrive to the system with rate λ^α (Poisson arrivals) and require service times with mean $1/\mu^\alpha$ (exponential). On the other hand, the low priority class β jobs arrive to the system with rate λ^β (Poisson arrivals) and require service times with mean $1/\mu^\beta$ (exponential). Jobs of both classes may be served at either of the two processors, which have service rates C_1 and C_2, respectively. Furthermore, for stability reasons it is assumed that the total arrival rate of service requirements is less than the total service rate : $\frac{\lambda^\alpha}{\mu^\alpha} + \frac{\lambda^\beta}{\mu^\beta} \leq C_1 + C_2$.

The fraction of class $c \in \{\alpha, \beta\}$ jobs assigned to server $i \in \{1,2\}$ is ϕ_i^c, where $\phi_1^c + \phi_2^c = 1, \ \ \phi_1^c, \phi_2^c \geq 0$, such that its cost function $J^c(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta)$ is minimized.

In the following sections, we formulate and solve the load sharing problem of two processors among two priority classes as a Stackelberg game. Due to space limitation, we omit the proofs which may be found in [5, 4].

3. STACKELBERG EQUILIBRIUM

In this section, we consider the load sharing problem, when two priority classes, with different objectives, share two processors. We formulate this priority multiobjective optimization problem as a non cooperative Stackelberg game [1] between the two priority classes.

Next, we give some definitions for a two-priority class (any kind of priorities) game similar to those in [1] for Stackelberg games:

Definition 1: *In a two-priority class finite game, with the high priority class α as the leader and the low priority class β as the follower, the set $R^\beta(\phi_1^\alpha, \phi_2^\alpha)$, defined for the high priority strategy $(\phi_1^\alpha, \phi_2^\alpha)$ that satisfies $\phi_1^\alpha + \phi_2^\alpha = 1, \ \ \phi_1^\alpha, \phi_2^\alpha \geq 0$, by:*

$$\begin{aligned} R^\beta(\phi_1^\alpha, \phi_2^\alpha) = \{ \ \ & (\phi_1^\beta, \phi_2^\beta) \ \ \textit{such that} \ \ \phi_1^\beta + \phi_2^\beta = 1, \ \ \phi_1^\beta, \phi_2^\beta \geq 0 : \\ & J^\beta(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta) \leq J^\beta(\phi_1^\alpha, \phi_2^\alpha, \hat{\phi}_1^\beta, \hat{\phi}_2^\beta), \\ & \forall \ (\hat{\phi}_1^\beta, \hat{\phi}_2^\beta), \ \ \textit{such that} \ \ \hat{\phi}_1^\beta + \hat{\phi}_2^\beta = 1, \ \ \hat{\phi}_1^\beta, \hat{\phi}_2^\beta \geq 0 \} \end{aligned}$$

is the optimal response (rational reaction) set of the low priority class β to the strategy of the high priority class α.

What the above definition says is that the low priority class β finds the set of its controls $(\phi_1^\beta, \phi_2^\beta)$, that minimize its cost function $J^\beta(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta)$, for given strategy $(\phi_1^\alpha, \phi_2^\alpha)$ of the high priority class α.

Definition 2: *In a two-priority class finite game with the high priority class α as the leader, a strategy $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, such that $\phi_1^{\alpha*} + \phi_2^{\alpha*} = 1$, $\phi_1^{\alpha*}, \phi_2^{\alpha*} \geq 0$, is called a Stackelberg equilibrium strategy for the leader if*

$$\inf_{(\phi_1^\beta,\phi_2^\beta)\in R^\beta(\phi_1^{\alpha*},\phi_2^{\alpha*})} J^\alpha(\phi_1^{\alpha*},\phi_2^{\alpha*},\phi_1^\beta,\phi_2^\beta) \leq \inf_{(\phi_1^\beta,\phi_2^\beta)\in R^\beta(\phi_1^\alpha,\phi_2^\alpha)} J^\alpha(\phi_1^\alpha,\phi_2^\alpha,\phi_1^\beta,\phi_2^\beta)$$

$\forall$ $(\phi_1^\alpha, \phi_2^\alpha)$ *such that* $\phi_1^\alpha + \phi_2^\alpha = 1$, $\phi_1^\alpha, \phi_2^\alpha \geq 0$.

This means that the high priority class α finds the set of its optimal controls $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$ that minimize its cost function $J^\alpha(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta)$, given the optimal response set $R^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*})$ of the low priority class β to its strategy $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$.

Definition 3: *Let $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, such that $\phi_1^{\alpha*} + \phi_2^{\alpha*} = 1$, $\phi_2^{\alpha*}, \phi_2^{\alpha*} \geq 0$, be a Stackelberg strategy for the leader α. Then any element $(\phi_1^{\beta*}, \phi_2^{\beta*}) \in R^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*})$ is an optimal strategy for the follower β that is in equilibrium with $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$. The strategy $(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{\beta*}, \phi_2^{\beta*})$ is a Stackelberg solution for the game with the high priority class α as the leader and the cost pair $J^\alpha(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{\beta*}, \phi_2^{\beta*})$, $J^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{\beta*}, \phi_2^{\beta*})$ is the corresponding Stackelberg equilibrium outcome.*

So, after the high priority class α has found its equilibrium strategy $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, then the optimal response set of the low priority class β is given by $R^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*})$. Any element $(\phi_1^{\beta*}, \phi_2^{\beta*}) \in R^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*})$ is an optimal strategy for the low priority class β.

4. PREEMPTIVE RESUME PRIORITY LOAD SHARING

In this section, we give a simple example for two preemptive resume priority classes of jobs that share two processors. When a high priority job is assigned to a processor, if there is another high priority job there, then it is put in the queue. If there are only low priority jobs there, then the low priority job is preempted and the high priority one starts been executing immediately. When all the high priority jobs have finished receiving service, then the low priority job that was preempted resumes and continues receiving service [8, 2]. The high priority class α (leader) assigns its jobs to the two processors, such that the average delay of its jobs is minimized. On the other hand, the low priority class β (follower) assigns its jobs to the two processors, such that the average delay of its jobs is minimized, after the high priority class α has optimally assigned its jobs. Thus a Stackelberg equilibrium is achieved.

4.1 General Two Class Solution

The cost function that we use for the high preemptive resume priority class α is its average job delay [8]: $J^\alpha(\phi_1^\alpha, \phi_2^\alpha) = \sum_{i=1}^{2} \phi_i^\alpha * \dfrac{\dfrac{1}{\mu^\alpha C_i}}{1 - \dfrac{\lambda^\alpha \phi_i^\alpha}{\mu^\alpha C_i}}$.

Similarly, the cost function for the low preemptive resume priority class β is its average job delay [8]: $J^\beta(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta) = \sum_{i=1}^{2} \phi_i^\beta * \dfrac{[\dfrac{1}{\mu^\beta C_i} - \dfrac{\lambda^\alpha \phi_i^\alpha}{\mu^\alpha C_i * \mu^\beta C_i} + \dfrac{\lambda^\alpha \phi_i^\alpha}{(\mu^\alpha C_i)^2}]}{(1 - \dfrac{\lambda^\alpha \phi_i^\alpha}{\mu^\alpha C_i}) * (1 - \dfrac{\lambda^\alpha \phi_i^\alpha}{\mu^\alpha C_i} - \dfrac{\lambda^\beta \phi_i^\beta}{\mu^\beta C_i})}$.

The overall average job delay is:

$$J(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta) = \frac{\lambda^\alpha}{\lambda^\alpha + \lambda^\beta} * J^\alpha(\phi_1^\alpha, \phi_2^\alpha) + \frac{\lambda^\beta}{\lambda^\alpha + \lambda^\beta} * J^\beta(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta)$$

Theorem 1 : There exists a Stackelberg equilibrium.

The high preemptive resume priority class α solves the following problem:

$$\textit{minimize} \qquad J^{\alpha}(\phi_1^{\alpha}, \phi_2^{\alpha}) = \sum_{i=1}^{2} \frac{\phi_i^{\alpha}}{\mu^{\alpha} C_i - \lambda^{\alpha} \phi_i^{\alpha}}$$

$$\begin{array}{ll} \textit{with respect to} & \phi_1^{\alpha},\ \phi_2^{\alpha} \\ \textit{such that} & \phi_1^{\alpha} + \phi_2^{\alpha} = 1, \quad \phi_1^{\alpha},\ \phi_2^{\alpha} \geq 0. \end{array}$$

On the other hand, the low preemptive resume priority class β solves the following problem:

$$\textit{minimize} \qquad J^{\beta}(\phi_1^{\alpha *}, \phi_2^{\alpha *}, \phi_1^{\beta}, \phi_2^{\beta}) = \sum_{i=1}^{2} \frac{\phi_i^{\beta} * [\dfrac{C_i}{\mu^{\beta}} - \dfrac{\lambda^{\alpha} \phi_i^{\alpha *}}{\mu^{\alpha} \mu^{\beta}} + \dfrac{\lambda^{\alpha} \phi_i^{\alpha *}}{(\mu^{\alpha})^2}]}{(C_i - \dfrac{\lambda^{\alpha} \phi_i^{\alpha *}}{\mu^{\alpha}}) * (C_i - \dfrac{\lambda^{\alpha} \phi_i^{\alpha *}}{\mu^{\alpha}} - \dfrac{\lambda^{\beta} \phi_i^{\beta}}{\mu^{\beta}})}$$

$$\begin{array}{ll} \textit{with respect to} & \phi_1^{\beta},\ \phi_2^{\beta} \\ \textit{such that} & \phi_1^{\beta} + \phi_2^{\beta} = 1, \quad \phi_1^{\beta},\ \phi_2^{\beta} \geq 0. \end{array}$$

Then, the following policy allocates the arriving jobs to the two servers such that a Stackelberg equilibrium is achieved:

$$\textit{If} \qquad \frac{\lambda^{\alpha}}{\mu^{\alpha}} \leq C_1 + C_2,$$

then

$$\textit{If} \quad C_1 - \sqrt{C_1 C_2} \leq \frac{\lambda^{\alpha}}{\mu^{\alpha}} \ \textit{and} \ C_2 - \sqrt{C_1 C_2} \leq \frac{\lambda^{\alpha}}{\mu^{\alpha}}$$

$$\textit{then} \ \ \phi_1^{\alpha *} = \frac{C_1}{\dfrac{\lambda^{\alpha}}{\mu^{\alpha}}} - \frac{C_1 + C_2 - \dfrac{\lambda^{\alpha}}{\mu^{\alpha}}}{\dfrac{\lambda^{\alpha}}{\mu^{\alpha}}} * \frac{\sqrt{C_1}}{\sqrt{C_1} + \sqrt{C_2}}$$

$$\textit{If} \quad 0 \leq \frac{\lambda^{\alpha}}{\mu^{\alpha}} \leq C_1 - \sqrt{C_1 C_2}$$

$$\textit{then} \ \ \phi_1^{\alpha *} = 1$$

$$\textit{If} \quad 0 \leq \frac{\lambda^{\alpha}}{\mu^{\alpha}} \leq C_2 - \sqrt{C_1 C_2}$$

$$\textit{then} \ \ \phi_1^{\alpha *} = 0$$

$$If \quad \frac{\lambda^\beta}{\mu^\beta} \le C_1^\alpha + C_2^\alpha$$

then

$$If \quad C_1^\alpha - C_2^\alpha * \sqrt{\frac{C_1^\beta}{C_2^\beta}} \le \frac{\lambda^\beta}{\mu^\beta} \quad and \quad C_2^\alpha - C_1^\alpha * \sqrt{\frac{C_2^\beta}{C_1^\beta}} \le \frac{\lambda^\beta}{\mu^\beta}$$

$$then \quad \phi_1^{\beta *} = \frac{C_1^\alpha}{\frac{\lambda^\beta}{\mu^\beta}} - \frac{C_1^\alpha + C_2^\alpha - \frac{\lambda^\beta}{\mu^\beta}}{\frac{\lambda^\beta}{\mu^\beta}} * \frac{\sqrt{C_1^\beta}}{\sqrt{C_1^\beta} + \sqrt{C_2^\beta}}$$

$$If \quad \frac{\lambda^\beta}{\mu^\beta} \le C_1^\alpha - C_2^\alpha * \sqrt{\frac{C_1^\beta}{C_2^\beta}}$$

$$then \quad \phi_1^{\beta *} = 1$$

$$If \quad \frac{\lambda^\beta}{\mu^\beta} \le C_2^\alpha - C_2^\alpha * \sqrt{\frac{C_2^\beta}{C_1^\beta}}$$

$$then \quad \phi_1^{\beta *} = 0$$

where

$$C_1^\alpha = C_1 - \frac{\lambda^\alpha \phi_1^{\alpha *}}{\mu^\alpha} \qquad C_2^\alpha = C_2 - \frac{\lambda^\alpha \phi_2^{\alpha *}}{\mu^\alpha}$$

$$C_1^\beta = \frac{C_1}{\mu^\beta} - \frac{\lambda^\alpha \phi_1^{\alpha *}}{\mu^\alpha \mu^\beta} + \frac{\lambda^\alpha \phi_1^{\alpha *}}{(\mu^\alpha)^2} \qquad C_2^\beta = \frac{C_2}{\mu^\beta} - \frac{\lambda^\alpha \phi_2^{\alpha *}}{\mu^\alpha \mu^\beta} + \frac{\lambda^\alpha \phi_2^{\alpha *}}{(\mu^\alpha)^2}$$

Of course, the equilibrium load sharing fractions to the other server are $\phi_2^{\alpha *} = 1 - \phi_1^{\alpha *}$ and $\phi_2^{\beta *} = 1 - \phi_1^{\beta *}$. Substituting these equilibrium fractions into the average delay functions, we have the equilibrium outcome of the game [5, 4]. Note also, that the Stackelberg solution is the same to the Nash solution for this problem, since the cost function of the leader is independent to the decisions of the follower.

4.2 Interesting Results

From the above equilibrium solution and outcome of the game, we have some interesting results:

Proposition 1: *Conservation law*
For a given system $C_1 \ge C_2$,
if $\mu^\alpha = \mu^\beta = \mu$, $\lambda^\alpha + \lambda^\beta \le \mu(C_1 + C_2)$, *and* $\lambda^\alpha + \lambda^\beta = \lambda = constant$,
then $J(\phi_1^{\alpha *}, \phi_2^{\alpha *}, \phi_1^{\beta *}, \phi_2^{\beta *}) = constant$

The above proposition says that for equal mean service requirements for both priority classes and constant total arrival rate, the overall average job delay is constant, i.e. it does not depend on the mix of high and low priority jobs. This result extends Kleinrock's conservation law for single queues [8].

In Fig. 2, we show the equilibrium average delay of the high priority class α, $J^\alpha(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, of the low priority class β, $J^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{\beta*}, \phi_2^{\beta*})$, and of the system $J(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{\beta*}, \phi_2^{\beta*})$ versus different mixes of the high and low priority arrival rates $\frac{\lambda^\alpha}{\lambda^\beta}$, for fixed server capacities $C_1 = 2, C_2 = 1$, fixed total arrival rate $\lambda^\alpha + \lambda^\beta = 2.5$, and equal mean service requirement of the high and the low priority jobs $1/\mu^\alpha = 1/\mu^\beta = 1$. We note that the overall average job delay is constant and independent from the mix of the high and low priority jobs.

Proposition 2: *For a given system with $C_1 \geq C_2$ and $\mu^\alpha = \mu^\beta = \mu$,*
if $\mu(C_1 + C_2) \leq \lambda^\alpha + \lambda^\beta$ and $C_1 - \sqrt{C_1 C_2} \leq \lambda^\alpha/\mu$,
then $\phi_1^{\beta*} = \dfrac{\sqrt{C_1}}{\sqrt{C_1} + \sqrt{C_2}}$

What the above proposition says is that for equal mean service requirements for both priority classes, when the high priority class α uses both servers, then the equilibrium decisions of the low priority class β are constant and independent of the arrival rates $(\lambda^\alpha, \lambda^\beta)$. This result is not intuitive, because we might expect that the load sharing decisions for the low priority class β should also depend on the arrival rates (as it is the case for the high priority class α). It is also very important, because even when the arrival rates vary over time, the load sharing policy for the low priority jobs remains the same (Fig. 3).

Proposition 3: *For a given system $C_1 \geq C_2$:*
if $\dfrac{\lambda^\alpha}{\mu^\alpha} + \dfrac{\lambda^\beta}{\mu^\beta} \to C_1 + C_2$, *then* $\phi_1^{\beta*} \to \dfrac{\sqrt{C_1}}{\sqrt{C_1} + \sqrt{C_2}}$.

The above proposition says that even for different service requirements for the two priority classes, when the total arriving service requirement $\frac{\lambda^\alpha}{\mu^\alpha} + \frac{\lambda^\beta}{\mu^\beta}$ approaches the total service capacity $C_1 + C_2$, the load sharing decisions for the low priority class β become constant and independent from the arrival rates.

In Fig. 3, we show the equilibrium load sharing fractions of both the high priority class α, $\phi_1^{\alpha*}$, and of the low priority class β, $\phi_1^{\beta*}$, versus the system load $\frac{\lambda^\alpha/\mu^\alpha + \lambda^\beta/\mu^\beta}{C_1 + C_2}$, for fixed server capacities $C_1 = 2, C_2 = 1$, equal arrival rates $\lambda^\alpha = \lambda^\beta$ and different ratio of the mean service requirement of the high and the low priority jobs $1/\mu^\alpha = 2/\mu^\beta = 1$, $1/\mu^\alpha = 1/\mu^\beta = 1$, $1/\mu^\beta = 2/\mu^\alpha = 1$.

For equal mean service requirements of the high and low priority jobs, we note that when the high priority class α uses both processors $(0 < \phi_1^{\alpha*} < 1)$, then the load sharing decisions $\phi_1^{\beta*}$ for the low priority class β are constant and independent of the arrival rates $\lambda^\alpha, \lambda^\beta$. That means that even if the arrival rates change during operation, our load sharing algorithm will still perform "optimally" for the low priority class. For different mean service requirements of the high and low priority jobs, we note that when the system load $\frac{\lambda^\alpha/\mu^\alpha + \lambda^\beta/\mu^\beta}{C_1 + C_2}$ approaches 1, then the load sharing decisions $\phi_1^{\beta*}$ for the low priority class β approach the same constant value as for the case of equal mean service requirement.

5. NUMERICAL RESULTS & DISCUSSION

In this section, we discuss some other results that can be derived from the solution of the two processor load sharing problem among jobs from two preemptive resume priority classes.

5.1 Constant λ^α

Consider a two processor system $C_1 \leq C_2$ with fixed arrival rate of interactive (high priority) jobs $\lambda^\alpha = \textit{constant}$. If this multiprocessor is also to be used by batch (low priority) jobs and we want to secure an upper bound on the average delay of batch jobs $J^\beta \leq J_0^\beta$, then we should restrict the arrival rate of the batch jobs up to an upper limit. For example, if $\mu^\alpha = \mu^\beta = \mu$, Case 1, then

$$\frac{(\sqrt{C_1}+\sqrt{C_2})^2}{\mu * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu}) * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu} - \frac{\lambda^\beta}{\mu})} \leq J_0^\beta \quad \Rightarrow$$

$$\lambda^\beta \leq \mu * (C_1 + C_2) - \lambda^\alpha - \frac{(\sqrt{C_1}+\sqrt{C_2})^2}{J_0^\beta * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu})}$$

In Fig. 4, we show the equilibrium average delay of both the higher priority class α, $J^\alpha(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, of the lower priority class β, $J^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{*}, \phi_2^{\beta*})$, and of the system $J(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{*}, \phi_2^{\beta*})$ versus the low priority class β arrival rate, λ^β, for fixed server capacities $C_1 = 2, C_2 = 1$, fixed mean service requirements $1/\mu^\alpha = 1/\mu^\beta = 1$ and fixed high priority class α arrival rate $\lambda^\alpha = 1.0$. So, for example, if the average delay of batch jobs J^β should be less than 10, then the arrival rate of batch jobs should be $\lambda^\beta < 1.71$.

5.2 Constant λ^β

Next, consider a two processor system $C_1 \leq C_2$ with fixed arrival rate of batch jobs (low priority) $\lambda^\beta = \textit{constant}$. If this multiprocessor is also to be used by interactive jobs (high priority) and we want to secure an upper bound on the average delay of batch jobs $J^\beta \leq J_0^\beta$, then we should restrict the arrival rate of the interactive jobs up to an upper limit. For example, if $\mu^\alpha = \mu^\beta = \mu$, Case 1, then

$$\frac{(\sqrt{C_1}+\sqrt{C_2})^2}{\mu * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu}) * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu} - \frac{\lambda^\beta}{\mu})} \leq J_0^\beta \quad \Rightarrow$$

$$\lambda^\alpha \leq \mu(C_1 + C_2) - \frac{\lambda^\beta}{2} - \sqrt{(\frac{\lambda^\beta}{2})^2 + \frac{\mu(\sqrt{C_1}+\sqrt{C_2})^2}{J_0^\beta}}$$

In Fig. 5, we show the equilibrium average delay of both the higher priority class α, $J^\alpha(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, of the lower priority class β, $J^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{*}, \phi_2^{\beta*})$, and of the system $J(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{*}, \phi_2^{\beta*})$ versus the high priority class α arrival rate, λ^α, for fixed server capacities $C_1 = 2, C_2 = 1$, fixed mean service requirements $1/\mu^\alpha = 1/\mu^\beta = 1.$ and fixed low priority class β arrival rate $\lambda^\beta = 1.0$.

So, for example, if the average delay of batch jobs J^β should be less than 10, then the arrival rate of interactive jobs should be $\lambda^\beta < 1.59$.

5.3 Constant λ

Thirdly, consider a two processor system $C_1 \leq C_2$ with fixed total arrival rate of both interactive and batch jobs $\lambda^{\alpha} + \lambda^{\beta} = \lambda = constant$. We want to determine what mix of interactive and batch jobs will secure an upper bound on the average delay of interactive jobs $J^{\leq} J_0^{\alpha}$ as well as on batch jobs $J^{\beta} \leq J_0^{\beta}$. Let $k = \frac{\lambda^{\alpha}}{\lambda^{\beta}}$ be the mix of interactive over batch jobs. Then we can write the arrival rate of interactive jobs as $\lambda^{\alpha} = \frac{k}{k+1}\lambda$ and the arrival rate of batch jobs as $\lambda^{\beta} = \frac{1}{k+1}\lambda$. For example, if $\mu^{\alpha} = \mu^{\beta} = 1$, Case 2.2, then

$$\frac{1}{\mu C_1 - \frac{k}{k+1}\lambda} \leq J_0^{\alpha} \qquad \Rightarrow k \leq \frac{J_0^{\alpha} * \mu C_1}{1 - J_0^{\alpha} * (\mu C_1 - \lambda)}$$

$$\frac{\mu C_1}{(\mu C_1 - \frac{k}{k+1}\lambda) * (\mu C_1 - \lambda)} \leq J_0^{\beta} \qquad \Rightarrow k \leq \frac{\mu C_1 * [J_0^{\beta} * (\mu C_1 - \lambda) - 1]}{\mu C_1 - J_0^{\beta} * (\mu C_1 - \lambda)^2}$$

In Fig. 2, we show the equilibrium average delay of both the high priority class α, $J^{\alpha}(\phi_1^{\alpha *}, \phi_2^{\alpha *})$, of the low priority class β, $J^{\beta}(\phi_1^{\alpha *}, \phi_2^{\alpha *}, \phi_1^{*}, \phi_2^{\beta *})$, and of the system $J(\phi_1^{\alpha *}, \phi_2^{\alpha *}, \phi_1^{*}, \phi_2^{\beta *})$ versus different mix of the high and low priority arrival rates $k = \frac{\lambda^{\alpha}}{\lambda^{\beta}}$, for fixed server capacities $C_1 = 2, C_2 = 1$, fixed total arrival rate $\lambda = \lambda^{\alpha} + \lambda^{\beta} = 2.5$, and equal mean service requirement of the high and the low priority jobs $1/\mu^{\alpha} = 1/\mu^{\beta} = 1$.

So, for example, if the average delay of interactive jobs should be less than 2 and the average delay of batch jobs should be less than 10, then mix of interactive and batch jobs should be $\frac{\lambda^{\alpha}}{\lambda^{\beta}} < 2.8$.

5.4 Different Server Rate Ratios

Finally, in Fig. 6, we show the equilibrium fraction to processor 1, of both the high and low priority classes versus the system load $\frac{\lambda^{\alpha}/\mu^{\alpha} + \lambda^{\beta}/\mu^{\beta}}{C_1 + C_2}$ for equal arrival rates $\lambda^{\alpha} = \lambda^{\beta}$, equal mean service requirements $1/\mu^{\alpha} = 1/\mu^{\beta} = 1$ and different ration of the service rates of the two processors $\frac{C_1}{C_2} = 5, 4, 3, 2, 1, 1/2, 1/3, 1/4, 1/5$.

When the service rate of server 1 is substantially larger than the service rate of server 2, $(C_1 = 5 * C_2)$, then server 1 is used exclusively for almost all arrival rates. When the service rates are $C_1 = 4 * C_2$, then for low and medium load, server 1 is exclusively used, but for heavy system load, server 2 also is used. When the service rates are $C_1 = 3 * C_2$, then the slow server starts been used for lower system load. When the service rates are $C_1 = 2 * C_2$, then the slow server starts been used for even lower system load. When the service rates are equal $C_1 = C_2$, then both servers are used equally $(\phi_1^{\alpha *} = \phi_2^{\alpha *} = \phi_1^{\beta *} = \phi_2^{\beta *} = 0.5)$. Now, when server 2 is faster, a similar scenario happens, i.e. the faster server 2 is, the more it is exclusively used.

6. CONCLUSIONS

In this paper, we formulate and solve a *priority load sharing* problem. Real distributed systems assign different priorities to different classes of jobs, in order to give preferential treatment to some classes of jobs. Therefore, it is not meaningful to optimize a single function over all different priority classes simultaneously. In this paper, we introduce an alternative methodology for dealing with multipriority optimization problems. We formulate a two-priority class load sharing problem as a Stackelberg game with leader the high priority class and follower the low priority class. Furthermore, we give the explicit solution when two preemptive resume priority classes want to minimize their average job delay. We find that for equal mean service requirements of jobs from both classes, when both processors are used, then the decisions of the low priority class do not depend on the arrival rates, i.e. even if the arrival rates vary, the same routing fractions can be used for the low priority jobs. Also, for equal mean service requirements of jobs from both classes, when the total arrival rate of jobs is constant but the mix of high and low priority jobs varies, then the overall average job delay remains constant.

Straightforward extensions are to consider multiple priority (> 2) classes, as well as more than two servers. This is the first study that formulates and solves a multi-priority resource allocation problem as a Stackelberg game. We have also formulated and solved the joint load sharing, routing and congestion control problem as a Nash and a Stackelberg game [4].

References

[1] T. Basar and G.J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, 1982.

[2] D.P. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, 1987.

[3] D.L. Eager, E.D. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Trans. on Software Eng., Vol SE-12, No 5*, pp. 662-675, May 1986.

[4] A.A. Economides. A unified game-theoretic methodology for the joint load sharing, routing and congestion control problem. *Ph.D. Dissertation, University of Southern California*, August 1990.

[5] A.A. Economides and J.A. Silvester. Priority load sharing: an approach using Stackelberg games. *USC Technical Report CENG 89-39*, 1989.

[6] A.A. Economides and J.A. Silvester. Routing games. *USC Technical Report CENG 89-38*, 1989.

[7] A.A. Economides and J.A. Silvester. A game theory approach to cooperative and noncooperative routing problems. *Proc. IEEE International Telecommunications Symposium*, Brazil, Sept. 3-6, 1990.

[8] L. Kleinrock. *Queueing Systems, Vol. 2: Applications.* J. Wiley & Sons, 1976.

[9] Y.-T. Wang and R.T.J. Morris. Load sharing in distributed systems. *IEEE Trans. on Computers, Vol C-34, No 3*, pp. 204-217, March 1985.

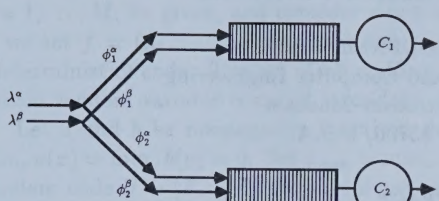


Figure 1 Two-class two-processor load sharing.

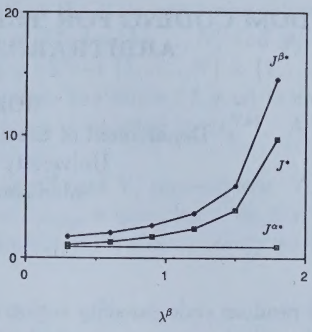


Figure 4 : Stackelberg equilibrium average delay J^α, J^β and J versus λ^β, for $C_1 = 2, C_2 = 1$, $\mu^\alpha = \mu^\beta = 1$ $\lambda^\alpha = 1.0$.

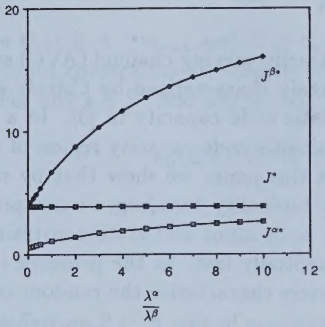


Figure 2 Stackelberg equilibrium average delay J^α, J^β and J versus different mixes of the high and low priority arrival rates $\frac{\lambda^\alpha}{\lambda^\beta}$, for $C_1 = 2, C_2 = 1$, $\lambda^\alpha + \lambda^\beta = 2.5$, and $\mu^\alpha = \mu^\beta = 1$.

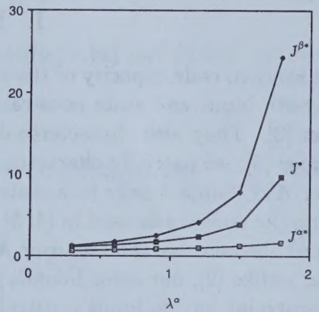


Figure 5 Stackelberg equilibrium average delay J^α, J^β and J versus λ^α, for $C_1 = 2, C_2 = 1$, $\mu^\alpha = \mu^\beta = 1$, and $\lambda^\beta = 1.0$.

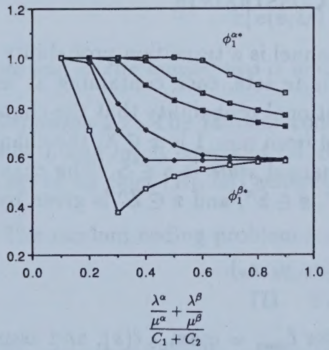


Figure 3: Stackelberg equilibrium fractions $\phi_1^{\alpha*}$ and $\phi_1^{\beta*}$ versus $\frac{\lambda^\alpha/\mu^\alpha + \lambda^\beta/\mu^\beta}{C_1 + C_2}$, for $C_1 = 2, C_2 = 1, \lambda^\alpha = \lambda^\beta$ and $\mu^\alpha = 2 * \mu^\beta = 1$, $\mu^\alpha = \mu^\beta = 1$ and $2 * \mu^\alpha = \mu^\beta = 1$.

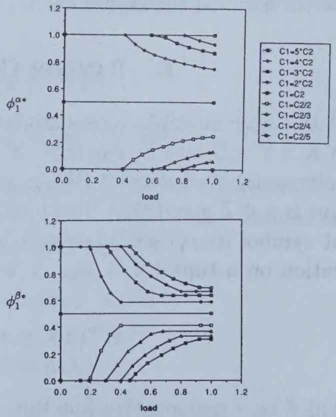


Figure 6 : Stackelberg equilibrium fraction to processor 1, of both the high and low priority classes versus the system load $\frac{\lambda^\alpha/\mu^\alpha + \lambda^\beta/\mu^\beta}{C_1 + C_2}$ for equal arrival rates $\lambda^\alpha = \lambda^\beta$, equal mean service requirements $1/\mu^\alpha = 1/\mu^\beta = 1$ and different ratio of the service rates of the two processors $\frac{C_1}{C_2} = 5, 4, 3, 2, 1, 1/2, 1/3, 1/4, 1/5$.

RANDOM CODING FOR THE CONSTRAINED MULITPLE-ACCESS ARBITRARILY VARYING CHANNEL

JOHN A. GUBNER
Department of Electrical and Computer Engineering
University of Wisconsin–Madison
Madison, WI 53706, U.S.A.

ABSTRACT

The random code capacity region of the two-user multiple-access arbitrarily varying channel subject to a state constraint is completely characterized. The forward result is more general and is proved for channels subject to both input and state constraints.

I. INTRODUCTION

The random code capacity of the single-user arbitrarily varying channel (AVC) subject to both input and state constraints was completely characterized by Csiszár and Narayan [2]. They also characterized the deterministic code capacity in [3]. In a recent paper [5], we partially characterized the deterministic code capacity region of the two-user AVC subject only to a state constraint. In this paper we show that by randomizing the codewords used in [4, 5] and using only typicality decoding, we can prove a forward theorem for the two-user AVC subject to both input and state constraints. Further, unlike [2], our error bounds decrease exponentially fast. In the presence of a state constraint but no input constraints, we completely characterize the random code capacity region.

The paper is organized as follows. In Section II we introduce our notation and define the problem to be studied. In Section III we state our results, and in Section IV we prove our forward theorem.

II. RANDOM CODING UNDER CONSTRAINTS

A two-user multiple-access arbitrarily varying channel is a transition probability W from $\mathsf{X} \times \mathsf{Y} \times \mathsf{S}$ into Z, where X, Y, S, and Z are finite sets, each containing at least two elements. We interpret $W(z|x,y,s)$ as the conditional probability that the channel output is $z \in \mathsf{Z}$ given that the channel input symbol from user 1 is $x \in \mathsf{X}$, the channel input symbol from user 2 is $y \in \mathsf{Y}$, and that the channel *state* is $s \in \mathsf{S}$. The channel operation on n-tuples $\mathbf{x} = (x_1, \ldots, x_n) \in \mathsf{X}^n$, $\mathbf{y} \in \mathsf{Y}^n$, $\mathbf{s} \in \mathsf{S}^n$, and $\mathbf{z} \in \mathsf{Z}^n$ is given by

$$W^n(\mathbf{z}|\mathbf{x},\mathbf{y},\mathbf{s}) \triangleq \prod_{k=1}^{n} W(z_k|x_k, y_k, s_k).$$

Let ℓ be a nonnegative function defined on S. Set $\ell_{\max} = \max_{s \in \mathsf{S}} \ell(s)$, and assume $\min_{s \in \mathsf{S}} \ell(s) = 0$. For $\mathbf{s} \in \mathsf{S}^n$, define the time average $\ell_n(\mathbf{s}) = (1/n)\sum_{k=1}^{n} \ell(s_k)$. Then the AVC W is said to be subject to state constraint L if the state-selection mechanism can generate only those state sequences that satisfy the time-average constraint $\ell_n(\mathbf{s}) \leq L$. Define

$$\mathsf{S}^n(L) = \{\mathbf{s} \in \mathsf{S}^n : \ell_n(\mathbf{s}) \leq L\}.$$

Note that if $L \geq \ell_{\max}$, then $\mathsf{S}^n(L) = \mathsf{S}^n$ and we say that the state constraint is inactive.

Let N, M, and n be positive integers. Let $\mathbf{x}_i \in \mathsf{X}^n$, $i = 1, \ldots, N$, and $\mathbf{y}_j \in \mathsf{Y}^n$, $j = 1, \ldots, M$, be given, and consider any mapping $\varphi : \mathsf{Z}^n \to \{1, \ldots, N\} \times \{1, \ldots, M\}$. If we set $f = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and $g = (\mathbf{y}_1, \ldots, \mathbf{y}_M)$, then the triple (f, g, φ) is called a (deterministic) code. The set of all codes $u = (f, g, \varphi)$ is denoted by $\mathcal{U}_n^{NM}$. A $\mathcal{U}_n^{NM}$-valued random variable is called a *random code.*

Let a and b be nonnegative functions defined on X and Y, respectively. Assume $\min_x a(x) = \min_y b(y) = 0$. Set $a_{\max} = \max_x a(x)$ and $b_{\max} = \max_y b(y)$. We say that a random code $U = (F, G, \Phi)$ satisfies input constraint (A, B) if almost surely we have

$$\frac{1}{n}\sum_{k=1}^{n} a(X_{i,k}) \;\leq\; A, \qquad i = 1, \ldots, N, \tag{2.1a}$$

and

$$\frac{1}{n}\sum_{k=1}^{n} b(Y_{j,k}) \;\leq\; B, \qquad j = 1, \ldots, M. \tag{2.1b}$$

Note that if $A \geq a_{\max}$ and $B \geq b_{\max}$, *all* codes satisfy (2.1a) and (2.1b). In this case, the input constraints are said to be inactive.

For any $\mathbf{s} \in \mathsf{S}^n$ and any $u \in \mathcal{U}_n^{NM}$, we let

$$e_{ij}(\mathbf{s}, u) \;\triangleq\; W^n(\{\mathbf{z} \in \mathsf{Z}^n : \varphi(\mathbf{z}) \neq (i,j)\} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}), \tag{2.2}$$

and we set

$$e(\mathbf{s}, u) \;\triangleq\; \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} e_{ij}(\mathbf{s}, u).$$

Definition 2.1: A pair of nonnegative real numbers, (R_1, R_2), is said to be *achievable* under state constraint L and input constraint (A, B) using random codes if, given any $0 < \lambda < 1$, for all sufficiently large n, whenever $N \leq \exp(nR_1)$ and $M \leq \exp(nR_2)$, there exists a $\mathcal{U}_n^{NM}$-valued random code $U = (F, G, \Phi)$ satisfying (2.1a) and (2.1b) almost surely and such that

$$\mathsf{E}[\,e(\mathbf{s}, U)\,] \;\leq\; \lambda \quad \text{for all } \mathbf{s} \in \mathsf{S}^n(L). \tag{2.3}$$

(Here and in the sequel, exp is understood as being to the base 2.)

Definition 2.2: The random code capacity region of the AVC W under state constraint L and input constaint (A, B) is denoted by $C_r(W, L, A, B)$ and is defined to be the set of all pairs (R_1, R_2) achievable in the sense of Definition 2.1.

The random coding problem is to characterize $C_r(W, L, A, B)$.

III. STATEMENT OF RESULTS

To present our results, we use the following notation. Let $\mathcal{D}(\mathsf{S})$ denote the set of all probability distributions on S. The sets $\mathcal{D}(\mathsf{X})$, $\mathcal{D}(\mathsf{X} \times \mathsf{Y})$, etc., are defined similarly. For all $P_{XYZ} \in \mathcal{D}(\mathsf{X} \times \mathsf{Y} \times \mathsf{Z})$, we define $I_{\mathsf{Y} \wedge \mathsf{Z}|\mathsf{X}}(P_{XYZ}) = I(Y \wedge Z|X)$, where the defining expression on the right is the usual average mutual information computed with the distribution indicated on the left. The quantities $I_{\mathsf{X} \wedge \mathsf{Z}|\mathsf{Y}}(P_{XYZ})$, $I_{\mathsf{XY} \wedge \mathsf{Z}}(P_{XYZ})$, etc., are defined similarly.

Next, suppose $p \in \mathcal{D}(\mathsf{X})$, $q \in \mathcal{D}(\mathsf{Y})$, and $r \in \mathcal{D}(\mathsf{S})$. We define a probability distribution on $\mathsf{X} \times \mathsf{Y} \times \mathsf{S} \times \mathsf{Z}$ by setting

$$(p \times q \times r \times W)(x, y, s, z) \ = \ p(x)q(y)r(s)W(z|x, y, s).$$

The marginals can be expressed in the same notation, e.g., $(p \times q \times rW)(x, y, z)$, where $(rW)(z|x, y) = \sum_s r(s)W(z|x, y, s)$.

Although it is a slight abuse of notation, for $r \in \mathcal{D}(\mathsf{S})$, we set $\ell(r) = \sum_s \ell(s)r(s)$. We then define

$$\mathcal{D}^L(\mathsf{S}) \ = \ \{r \in \mathcal{D}(\mathsf{S}) : \ell(r) \le L\}.$$

Let

$$I^L_{\mathsf{Y}\wedge\mathsf{Z}|\mathsf{X}}(p, q, W) \ = \ \inf_{r \in \mathcal{D}^L(\mathsf{S})} I_{\mathsf{Y}\wedge\mathsf{Z}|\mathsf{X}}(p \times q \times rW).$$

The quantities $I^L_{\mathsf{X}\wedge\mathsf{Z}|\mathsf{Y}}(p, q, W)$, $I^L_{\mathsf{XY}\wedge\mathsf{Z}}(p, q, W)$, etc., are defined similarly. With these definitions, we set

$$\begin{aligned} \mathcal{R}^L(p, q, W) = \{(R_1, R_2) : \ & 0 \le R_1 \le I^L_{\mathsf{X}\wedge\mathsf{Z}|\mathsf{Y}}(p, q, W), \\ & 0 \le R_2 \le I^L_{\mathsf{Y}\wedge\mathsf{Z}|\mathsf{X}}(p, q, W), \\ & R_1 + R_2 \le I^L_{\mathsf{XY}\wedge\mathsf{Z}}(p, q, W)\}. \end{aligned}$$

Just as with state constraints, we abuse notation and write $a(p) = \sum_x a(x)p(x)$ and $b(q) = \sum_y b(y)q(y)$. We also define

$$\mathcal{D}^A(\mathsf{X}) = \{p \in \mathcal{D}(\mathsf{X}) : a(p) \le A\} \quad \text{and} \quad \mathcal{D}^B(\mathsf{Y}) = \{q \in \mathcal{D}(\mathsf{Y}) : b(q) \le B\}.$$

Let $\mathcal{R}^{L,A,B}(W)$ denote the closed convex hull of

$$\bigcup_{p \in \mathcal{D}^A(\mathsf{X}),\, q \in \mathcal{D}^B(\mathsf{Y})} \mathcal{R}^L(p, q, W).$$

Our main result is the following theorem, whose proof is given in the next section.

Theorem 3.1: The random code capacity region of the two-user arbitrarily varying channel under state constraint L and input constraint (A, B) satisfies $\mathcal{R}^{L,A,B}(W) \subset C_r(W, L, A, B)$.

Corollary 3.2: If $A \ge a_{\max}$ and $B \ge b_{\max}$, then the input constraints are inactive and all codes satisfy (2.1a) and (2.1b). In this case, $C_r(W, L, A, B) = \mathcal{R}^{L,A,B}(W)$.

Proof: The fact that $\mathcal{R}^{L,A,B}(W) \subset C_r(W, L, A, B)$ follows from Theorem 3.1. In the case that $A \ge a_{\max}$ and $B \ge b_{\max}$, the fact that $C_r(W, L, A, B) \subset \mathcal{R}^{L,A,B}(W)$, is fairly standard and is therefore omitted (see, e.g., [5, Section III-A], especially the remark at the end of that section). □

IV. PROOF OF THEOREM 3.1

Our proof relies very heavily on the method of *types* as discussed in [1, pp. 29-33]. Recall that the *type* of an n-tuple $\mathbf{x} \in \mathsf{X}^n$ is defined to be the probability distribution $P_{\mathbf{x}}$ given by $P_{\mathbf{x}}(\alpha) = N(\alpha|\mathbf{x})/n$ for $\alpha \in \mathsf{X}$, where $N(\alpha|\mathbf{x})$ denotes the number of occurrences of α in the n-tuple $\mathbf{x}$. The set of types generated by X^n is denoted by $\mathcal{D}_n(\mathsf{X})$; more precisely, $\mathcal{D}_n(\mathsf{X})$ is the set of $P \in \mathcal{D}(\mathsf{X})$ such that $P = P_{\mathbf{x}}$ for some

$\mathbf{x} \in \mathsf{X}^n$. In an analogous way, the *joint type* of a pair of n-tuples, $\mathbf{x}$ and $\mathbf{y}$, is defined by $P_{\mathbf{x},\mathbf{y}}(\alpha,\beta) = N(\alpha,\beta|\mathbf{x},\mathbf{y})/n$ for $\alpha \in \mathsf{X}$ and $\beta \in \mathsf{Y}$, where $N(\alpha,\beta|\mathbf{x},\mathbf{y})$ denotes the number of occurrences of (α,β) in the n-tuple $((x_1,y_1),\dots,(x_n,y_n))$.

We next introduce what we call the ε–δ convention. By this we mean the following. All of the mutual informations defined at the beginning of Section III as well as similarly defined entropy functions used below are uniformly continuous functions of P_{XYZ}. Let $D(\cdot\|\cdot)$ denote the Kullback-Leibler informational divergence [1, p. 20, p. 58]. Given $\varepsilon > 0$, there exists a $\delta > 0$ such that whenever $D(P_{XYZ}\|Q_{XYZ}) \le \delta$, any mutual information or entropy evaluated at P_{XYZ} is within $\varepsilon/2$ of the same function evaluated at Q_{XYZ}.

By the usual time-sharing principle [1, p. 272], it suffices to prove that for all $p \in \mathcal{D}^A(\mathsf{X})$ and $q \in \mathcal{D}^B(\mathsf{Y})$, $\mathcal{R}^L(p,q,W) \subset C_r(W,L,A,B)$. Since the capacity region is closed, we may restrict attention to interior points of $\mathcal{R}^L(p,q,W)$. (The case in which $\mathcal{R}^L(p,q,W)$ has no interior points can be handled with minor modifications to the proof to follow.) Suppose (R_1,R_2) is an interior point of $\mathcal{R}^L(p,q,W)$. Then we can choose $\varepsilon > 0$ so small that $R_1 + 2\varepsilon < I^L_{\mathsf{X}\wedge\mathsf{Z}|\mathsf{Y}}(p,q,W)$, $R_2 + 2\varepsilon < I^L_{\mathsf{Y}\wedge\mathsf{Z}|\mathsf{X}}(p,q,W)$, and $R_1 + R_2 + 2\varepsilon < I^L_{\mathsf{XY}\wedge\mathsf{Z}}(p,q,W)$. Let δ be as in the ε–δ convention. Let n be sufficiently large; as will become clear in the course of the proof, how large n needs to be depends only on ε, δ, and the cardinalities of the sets X, Y, S, and Z. We begin by assuming n is large enough that we can approximate p and q by *types* $P \in \mathcal{D}_n(\mathsf{X})$ and $Q \in \mathcal{D}_n(\mathsf{Y})$ such that $I^L_{\mathsf{X}\wedge\mathsf{Z}|\mathsf{Y}}(P,Q,W) > I^L_{\mathsf{X}\wedge\mathsf{Z}|\mathsf{Y}}(p,q,W) - \varepsilon$, $I^L_{\mathsf{Y}\wedge\mathsf{Z}|\mathsf{X}}(P,Q,W) > I^L_{\mathsf{Y}\wedge\mathsf{Z}|\mathsf{X}}(p,q,W) - \varepsilon$, $I^L_{\mathsf{XY}\wedge\mathsf{Z}}(P,Q,W) > I^L_{\mathsf{XY}\wedge\mathsf{Z}}(p,q,W) - \varepsilon$, and such that $a(P) \le A$ and $b(Q) \le B$. For any $N \le \exp(nR_1)$ and $M \le \exp(nR_2)$, we will construct a random code U satisfying (2.1a) and (2.1b) and such that for all i,j and all $\mathbf{s} \in \mathsf{S}^n(L)$,

$$\begin{aligned} \mathsf{E}[\, e_{ij}(\mathbf{s},U)\,] \;\le\;& 3\exp(-n\delta/8) \\ &+ \exp[-n(I^L_{\mathsf{X}\wedge\mathsf{Z}|\mathsf{Y}}(p,q,W) - R_1 - 2\varepsilon)] \\ &+ \exp[-n(I^L_{\mathsf{Y}\wedge\mathsf{Z}|\mathsf{X}}(p,q,W) - R_2 - 2\varepsilon)] \\ &+ \exp[-n(I^L_{\mathsf{XY}\wedge\mathsf{Z}}(p,q,W) - R_1 - R_2 - 2\varepsilon)]. \end{aligned} \tag{4.1}$$

The Decoder

Given a set of codewords $f = (\mathbf{x}_1,\dots,\mathbf{x}_N)$ and $g = (\mathbf{y}_1,\dots,\mathbf{y}_M)$, we show how to construct a typicality decoder φ. In other words, we shall prescribe a deterministic function typ and set $\varphi = \text{typ}(f,g)$. If (F,G) is a random set of codewords, we can generate a random code $U = (F,G,\Phi)$ by taking $\Phi = \text{typ}(F,G)$.

For $\mathbf{s} \in \mathsf{S}^n$, let

$$K(\mathbf{s}) \;=\; \{(\mathbf{x},\mathbf{y},\mathbf{z}) : D(P_{\mathbf{x},\mathbf{y},\mathbf{s},\mathbf{z}}\|P \times Q \times P_{\mathbf{s}} \times W) \le \delta\},$$

and set

$$K \;=\; \bigcup_{\mathbf{s}' \in \mathsf{S}^n(L)} K(\mathbf{s}'). \tag{4.2}$$

Now, to define a decoder on Z^n, we use the sets $K_{ij} = \{\mathbf{z} : (\mathbf{x}_i,\mathbf{y}_j,\mathbf{z}) \in K\}$ as follows. Let

$$E_{ij} \;=\; K_{ij} \cap \Big(\bigcup_{i' \ne i} K_{i'j}\Big)^c \cap \Big(\bigcup_{j' \ne j} K_{ij'}\Big)^c \cap \Big(\bigcup_{i' \ne i,\, j' \ne j} K_{i'j'}\Big)^c. \tag{4.3}$$

Clearly, the $\{E_{ij}\}$ are disjoint. Let φ be any mapping such that $\mathbf{z} \in E_{ij}$ implies $\varphi(\mathbf{z}) = (i,j)$. This ensures that $\varphi(\mathbf{z}) \ne (i,j)$ implies $\mathbf{z} \in E_{ij}^c$.

Remainder of Proof

For any code $u \in \mathcal{U}_n^{NM}$ whose decoder is as just described, and for any $\mathbf{s} \in \mathsf{S}^n(L)$, we can always write

$$\begin{aligned} e_{ij}(\mathbf{s}, u) &\leq W^n(E_{ij}^c | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) \\ &\leq W^n(K_{ij}^c | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) && \text{(4.4a)} \\ &\quad + \sum_{i' \neq i} W^n(K_{i'j} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) && \text{(4.4b)} \\ &\quad + \sum_{j' \neq j} W^n(K_{ij'} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) && \text{(4.4c)} \\ &\quad + \sum_{i' \neq i,\, j' \neq j} W^n(K_{i'j'} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}). && \text{(4.4d)} \end{aligned}$$

We first examine the term in (4.4a). If we set $K_{ij}(\mathbf{s}) = \{\mathbf{z} : (\mathbf{x}_i, \mathbf{y}_j, \mathbf{z}) \in K(\mathbf{s})\}$, then $K_{ij}^c \subset K_{ij}(\mathbf{s})^c$. Using the method of types, it is then easy to show that for sufficiently large n (cf., e.g., [4, ineq. (A.19)]),

$$\begin{aligned} W^n(K_{ij}^c | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) &\leq W^n(K_{ij}(\mathbf{s})^c | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) \\ &= \sum_{\mathbf{z}:(\mathbf{x}_i, \mathbf{y}_j, \mathbf{z}) \in K(\mathbf{s})^c} W^n(\mathbf{z} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) && \text{(4.5a)} \\ &\leq \exp[-n(3\delta/4 - I(\mathbf{x}_i \wedge \mathbf{s}) - I(\mathbf{y}_j \wedge \mathbf{x}_i \mathbf{s}))], \end{aligned}$$

where $I(\mathbf{x}_i \wedge \mathbf{s})$ is shorthand for $I_{\mathsf{X} \wedge \mathsf{S}}(P_{\mathbf{x}_i, \mathbf{s}})$.

Now consider a term in the sum in (4.4b). We can write

$$W^n(K_{i'j} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) = \sum_{\mathbf{z}:(\mathbf{x}_{i'}, \mathbf{y}_j, \mathbf{z}) \in K} W^n(\mathbf{z} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}). \tag{4.5b}$$

Similarly, the terms in (4.4c) and (4.4d) can be written, respectively, as

$$W^n(K_{ij'} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) = \sum_{\mathbf{z}:(\mathbf{x}_i, \mathbf{y}_{j'}, \mathbf{z}) \in K} W^n(\mathbf{z} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}), \tag{4.5c}$$

and

$$W^n(K_{i'j'} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) = \sum_{\mathbf{z}:(\mathbf{x}_{i'}, \mathbf{y}_{j'}, \mathbf{z}) \in K} W^n(\mathbf{z} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}). \tag{4.5d}$$

To make further progress in bounding the probability of error, we use the following random code. Let

$$F = (\mathbf{X}_1, \ldots, \mathbf{X}_N) \quad \text{and} \quad G = (\mathbf{Y}_1, \ldots, \mathbf{Y}_M),$$

where F and G are independent. Letting (cf. [1, p. 29])

$$\mathsf{T}_P \triangleq \{\mathbf{x} \in \mathsf{X}^n : P_{\mathbf{x}} = P\} \quad \text{and} \quad \mathsf{T}_Q \triangleq \{\mathbf{y} \in \mathsf{Y}^n : P_{\mathbf{y}} = Q\},$$

we further take the $\{\mathbf{X}_i\}$ to be independent and uniformly distributed on T_P and the $\{\mathbf{Y}_j\}$ to be independent and uniformly distributed on T_Q. It is important to observe here that $\mathbf{x} \in \mathsf{T}_P$ implies

$$\begin{aligned} \frac{1}{n} \sum_{k=1}^{n} a(x_k) &= \sum_{x \in \mathsf{X}} a(x) P(x) \\ &= a(P) \\ &\leq A. \end{aligned}$$

Thus, all our random codewords satisfy (2.1a) and (2.1b).

Remark 4.1: In general outline, our proof is very similar to Jahn's proof of the forward result for the unconstrained two-user AVC [6, Section III-C]. However, a major difference is in the randomization of the codewords. Jahn used

$$\mathsf{P}(\mathbf{X}_i = \mathbf{x}) \;=\; \prod_{k=1}^{n} p(x_k) \qquad \text{and} \qquad \mathsf{P}(\mathbf{Y}_j = \mathbf{y}) \;=\; \prod_{k=1}^{n} q(y_k).$$

Note that having $a(p) \leq A$ and $b(q) \leq B$ does not imply that these random codewords will satisfy (2.1a) and (2.1b) almost surely. Our proof also differs from Jahn's in that our set K defined in (4.2) is somewhat different from Jahn's set K_ϵ defined at the top of [6, p. 220].

Let $e_a(\mathbf{x}_i, \mathbf{y}_j, \mathbf{s})$ denote the right-hand side of (4.5a). It is clearly upper bounded by 1. Hence,

$$\begin{aligned} \mathsf{E}[\, e_a(\mathbf{X}_i, \mathbf{Y}_j, \mathbf{s})\,] \;\leq\; & \exp(-n\delta/4) \\ & + \mathsf{P}\Big(\mathbf{X}_i \in \{\mathbf{x} : I(\mathbf{x} \wedge \mathbf{s}) > \delta/4\}\Big) \\ & + \mathsf{P}\Big((\mathbf{X}_i, \mathbf{Y}_j) \in \{(\mathbf{x}, \mathbf{y}) : I(\mathbf{y} \wedge \mathbf{x}\mathbf{s}) > \delta/4\}\Big). \end{aligned}$$

On account of the independence of $\mathbf{X}_i$ and $\mathbf{Y}_j$, an easy calculation using the method of types shows that these last two probabilities are each upper bounded by $\exp(-n\delta/8)$ for sufficiently large n.

Let $e_b(\mathbf{x}_i, \mathbf{y}_j, \mathbf{x}_{i'})$, $e_c(\mathbf{x}_i, \mathbf{y}_j, \mathbf{y}_{j'})$, and $e_d(\mathbf{x}_i, \mathbf{y}_j, \mathbf{x}_{i'}, \mathbf{y}_{j'})$ denote the right-hand sides of (4.5b)–(4.5d), respectively. Each is clearly upper bounded by 1. We treat only e_d as the others can be treated similarly. For $i' \neq i$ and $j' \neq j$, we can use independence to compute

$$\begin{aligned} & \mathsf{E}[\, e_d(\mathbf{X}_i, \mathbf{Y}_j, \mathbf{X}_{i'}, \mathbf{Y}_{j'}) \mid \mathbf{X}_i = \mathbf{x}_i,\, \mathbf{Y}_j = \mathbf{y}_j\,] && (4.6) \\ & \quad = \; \mathsf{E}[\, e_d(\mathbf{x}_i, \mathbf{y}_j, \mathbf{X}_{i'}, \mathbf{Y}_{j'})\,] \\ & \quad = \; \sum_{\mathbf{z} \in \mathsf{Z}^n} W^n(\mathbf{z} | \mathbf{x}_i, \mathbf{y}_j, \mathbf{s}) \Big(\sum_{(\mathbf{x},\mathbf{y}) : (\mathbf{x},\mathbf{y},\mathbf{z}) \in K} \mathsf{P}(\mathbf{X}_{i'} = \mathbf{x}, \mathbf{Y}_{j'} = \mathbf{y}) \Big). && (4.7) \end{aligned}$$

Since $\mathsf{P}(\mathbf{X}_{i'} = \mathbf{x}, \mathbf{Y}_{j'} = \mathbf{y}) = 1/(|\mathsf{T}_P|\,|\mathsf{T}_Q|)$, and since the cardinality of $\{(\mathbf{x}, \mathbf{y}) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in K\}$ is upper bounded by

$$(n+1)^{|\mathsf{X}||\mathsf{Y}||\mathsf{Z}|} \exp[n(H^L_{\mathsf{XY|Z}}(P, Q, W) + \varepsilon/2)],$$

where $H^L_{\mathsf{XY|Z}}(P, Q, W) = \sup_{r \in \mathcal{D}^L(\mathsf{S})} H_{\mathsf{XY|Z}}(P \times Q \times rW)$, the inner sum in (4.7) is upper bounded by $\exp[-n(I^L_{\mathsf{XY \wedge Z}}(P, Q, W) - \varepsilon)]$ when n is large enough. Since this bound does not depend on $\mathbf{z}$, the conditional expectation in (4.6) is upper bounded by the same quantity. Since the bound doesn't depend on $\mathbf{x}_i$ or $\mathbf{y}_j$,

$$\mathsf{E}[\, e_d(\mathbf{X}_i, \mathbf{Y}_j, \mathbf{X}_{i'}, \mathbf{Y}_{j'})\,] \;\leq\; \exp[-n(I^L_{\mathsf{XY \wedge Z}}(P, Q, W) - \varepsilon)]$$

as well.

By a completely analogous procedure, one can show that

$$\mathsf{E}[\, e_b(\mathbf{X}_i, \mathbf{Y}_j, \mathbf{X}_{i'})\,] \;\leq\; \exp[-n(I^L_{\mathsf{X \wedge Z|Y}}(P, Q, W) - \varepsilon)]$$

and

$$\mathsf{E}[\, e_c(\mathbf{X}_i, \mathbf{Y}_j, \mathbf{Y}_{j'})\,] \leq \exp[-n(I^L_{\mathsf{Y}\wedge\mathsf{Z}|\mathsf{X}}(P, Q, W) - \varepsilon)].$$

From (4.4a)-(4.4d), we now see that

$$\begin{aligned}\mathsf{E}[\, e_{ij}(\mathbf{s}, U)\,] \;\leq\; & 3\exp(-n\delta/8) \\ & + N\exp[-n(I^L_{\mathsf{X}\wedge\mathsf{Z}|\mathsf{Y}}(P, Q, W) - \varepsilon)] \\ & + M\exp[-n(I^L_{\mathsf{Y}\wedge\mathsf{Z}|\mathsf{X}}(P, Q, W) - \varepsilon)] \\ & + NM\exp[-n(I^L_{\mathsf{XY}\wedge\mathsf{Z}}(P, Q, W) - \varepsilon)],\end{aligned}$$

from which (4.1) follows. □

V. CONCLUSIONS

We have completely characterized the random code capacity region of the two-user AVC subject to a state constraint (but no input constraints). Our forward theorem, which holds when both input and state constraints are imposed on the channel, gives exponential bounds on the probability of error.

Work is currently in progress to obtain a converse theorem for the two-user AVC when both input and state constraints are active.

REFERENCES

[1] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems.* New York: Academic, 1981.

[2] I. Csiszár and P. Narayan, "Arbitrarily varying channels with constrained inputs and states," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 27-34, Jan. 1988.

[3] ——, "The capacity of the arbitrarily varying channel revisited: Positivity, constraints," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 181-193, Mar. 1988.

[4] J. A. Gubner, "On the deterministic-code capacity of the multiple-access arbitrarily varying channel," *IEEE Trans. Inform. Theory*, vol. IT-36, no. 2, pp. 262-275, Mar. 1990.

[5] ——, "State constraints for the multiple-access arbitrarily varying channel," *IEEE Trans. Inform. Theory*, to appear, vol. IT-37, Jan. 1991.

[6] J.-H. Jahn, "Coding of arbitrarily varying multiuser channels," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 212-226, Mar. 1981.

The Asymptotic Capacity of the Direct Detection Photon Channel With a Bandwidth Constraint

David Brady
Department of Electrical and
Computer Engineering
Northeastern University
Boston, MA 02115

Sergio Verdú
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

1. Abstract

In this paper we address the capacity of a direct detection photon counting channel with an average and symbol period constraint on the input. This channel model is ideally suited for, but not limited to, a communication system comprised of a semiconductor laser with electro-optic modulation, a fiber optic transmission medium, and an integrate-and-dump receiver. We find the limiting form of the capacity as the signal and noise energies increase proportionally and we show that the capacity grows logarithmically with the signal energy constraint.

2. Introduction

Consider a discrete-time memoryless channel with a nonnegative input $\mathcal{X}_i$ which is constrained in peak, $\mathcal{X}_i \leq M$, and sample average, $\bar{\mathcal{X}} \leq \sigma M$. The output of this channel is a conditionally Poisson count $\mathcal{Y}_i$ given the input $\mathcal{X}_i$. Under this conditioning $\mathcal{Y}_i$ has mean $\mathcal{X}_i + d$, where d represents the average count due to additive, independent noise. This channel is called the pulsewidth-limited direct detection photon channel (PLPC) and is a suitable model for a fiber optic communication channel with photoelectric conversion at the receiver, especially when semiconductor laser sources and long-haul networks are used. With respect to this continuous-time application, the peak input constraint M represents the product of the peak received optical power A and the fixed symbol period Δ, σ is the ratio of the average laser power to the peak laser power, and $\lambda \overset{\Delta}{=} \frac{d}{\Delta}$ is the dark current intensity at the photodetector. In practical applications the fixed symbol period Δ is an important constraint, and represents the limitations on the rate of communication imposed by electro-optic modulation, fiber dispersion, and other system anomolies. The PLPC is a suitable model for the fiber optic channel provided that the temporal information about the photons (or photoelectrons) is ignored, and the output is restricted to the count in each subinterval. Since complete point process observations are difficult to achieve in practice, and subinterval counting may be implemented by integrating the photocurrent, this restriction represents a technological limitation. Note that the channel input is distorted in two ways: by the additive noise due to dark current, and by the inevitable quantum or self-noise due to the statistical nature of electromagnetic radiation.

It is of interest to find the capacity of this channel for arbitrary values of A, σ, λ and Δ. Unfortunately, the capacity expression does not have an explicit form, due in part to the lack of a closed form for the entropy of the Poisson distribution. Instead, previous work has addressed the asymptotic capacity, as some of the constraints vanish. By ignoring the modulation rate limitation, for example, it has been shown that the capacity of the PLPC (in nats per second) is given by [1], [2], [3], [4]

$$\lim_{\Delta\to 0}\frac{C}{\Delta}=\begin{cases}\sigma^*(A+\lambda)\log(1+\frac{\lambda}{A})+(1-\sigma^*)\lambda\log(\frac{\lambda}{A}), & \sigma^*\le\sigma\\ \sigma(A+\lambda)\log(1+\frac{\lambda}{A})+(1-\sigma)\lambda\log(\frac{\lambda}{A}), & \sigma^*>\sigma\end{cases} \tag{1}$$

where $\sigma^* A$ is the optimum average input for the simpler problem when $\sigma = 1$, i.e., the average input constraint is inactive, and σ^* is given by

$$\sigma^*=\frac{\left(1+\frac{\lambda}{A}\right)^{1+\frac{\lambda}{A}}}{\left(\frac{\lambda}{A}\right)^{\frac{\lambda}{A}}\mathrm{e}}-\frac{\lambda}{A}.$$

In terms of the PLPC constraints, this asymptotic result is equivalent to $M \to 0$ or $\bar{\mathcal{X}} \to 0$ with fixed $SNR \triangleq \bar{\mathcal{X}}/d$. It is interesting to note that as $\Delta \to 0$, the PLPC channel capacity coincides with that of a channel whose output is the complete point process observation. This fact is due to the conditional orderliness, or "non-explosiveness" property of the double-stochastic point process. Since the PLPC restricts the information of each photoelectron arrival time to one of the Δ-width subintervals, conditional orderliness assures that as $\Delta \to 0$, at most one photoelectron arrives in each subinterval, and the arrival time of each is known in this limit.

Several works have also addressed the limiting form of the PLPC capacity for a nontrivial modulation constraint Δ. A lower bound to the PLPC capacity was investigated for an inactive peak constraint, $d = 0$, and $\bar{\mathcal{X}} \to \infty$ [5] . These restrictions represent the case in which the quantum model of electromagnetic propagation approaches the continuous model, and both the quantum and dark current noise may be neglected. It was shown that the capacity of the noiseless PLPC grows at least as fast as $\log \sqrt{\bar{\mathcal{X}}}$

$$\lim_{\substack{\bar{\mathcal{X}} \to \infty \\ SNR=\infty}} \frac{C}{\log \sqrt{\frac{\bar{\mathcal{X}}}{2\pi}}} \geq 1.$$

A lower bound to the PLPC capacity with an inactive average constraint was investigated by considering discrete input distributions having K equally-spaced, equally-likely points of mass on $[0, M]$ [6] . For fixed M and d, the mutual information was optimized numerically over K. A lower bound to the PLPC capacity was derived for $d = 0$ and large M

$$\lim_{\substack{M \to \infty \\ SNR=\infty}} \frac{C}{\log \sqrt{\frac{M}{2\pi}}} \geq 1. \tag{2}$$

In the absence of dark current, it was noted that the optimal spacing between the mass points was roughly $2.7\sqrt{M}$ for large M.

It has been shown recently that the capacity of the PLPC with an active peak constraint is achieved by a discrete input distribution having a finite set of mass points on $[0, M]$ [7]. This was demonstrated by arguments paralleling those of Smith [8] for the peak-constrained, additive Gaussian noise channel. It was proven that a binary input distribution maximizes the mutual information between channel input and output, not only as $M \to 0$, but also for $0 < M < \phi$, where ϕ is strictly positive. It was shown that equally-spaced, uniform, discrete input distributions do not maximize the mutual information between input and output.

In this paper we present the limiting form of the PLPC capacity for an inactive peak constraint, and for sufficiently large $\bar{\mathcal{X}}$. Since $M = \infty$, the results from [7] do not apply, and it is not known whether a discrete input distribution maximizes $\mathcal{I}(\mathcal{X}_i; \mathcal{Y}_i)$. Unlike previous works with an inactive peak constraint, we consider the practical case when the dark current cannot be neglected. In fact, we will be interested in the case when d may dominate the signal energy, $\bar{\mathcal{X}}$. We develop bounds on the PLPC capacity for increasing $\bar{\mathcal{X}}$ and constant SNR, and show that these bounds tighten as $\bar{\mathcal{X}} \to \infty$. Since d and $\bar{\mathcal{X}}$ are average counts, this represents a situation in which the signal and noise counts grow at the same rate, either by increasing the subinterval length or by lowering the carrier frequency of the system (and the energy per photon). Although this is a case in which the semi-classical model approaches the classical model, we should not expect the PLPC capacity to approach the (finite) capacity of the discrete time additive Gaussian noise channel with an energy constraint. In fact, it will be demonstrated that the PLPC capacity is bounded for sufficiently large $\bar{\mathcal{X}}$ by

$$K_l \le \mathcal{C} - \log\sqrt{\frac{\bar{\mathcal{X}}}{2\pi}} \le K_u$$

where K_l, K_u are constants that depend only on SNR. This demonstrates that the lower bound on the asymptotic capacity with $SNR = \infty$ derived in [5] is also a lower bound to the (smaller) capacity of the PLPC with arbitrarily large dark current, λ. More importantly, we demonstrate that the lower bound is tight. The positive growth of the PLPC capacity for fixed SNR is not intuitive, and suggests that the Poisson noise count is "less random" as the noise and signal energies increase proportionally. This result is an artifact of the Poisson distribution, and complements previous work on singular detection in additive Poisson noise [9].

3. Analysis

Throughout this section Π denotes a Poisson random variable with mean m. It will be demonstrated presently that the expectation of $\log \Pi!$ coincides with $\log m!$ for large m by using Stirling's bounds to $k!$

$$\lfloor k \rceil ! \triangleq \sqrt{2\pi k}\left(\frac{k}{\mathrm{e}}\right)^k \le k! \le \lfloor k \rceil !\left(1 + \frac{1}{12k-1}\right) \tag{3}$$

A reciprocal moment of the Poisson random variable will also be used in this demonstration. While reciprocal moments are difficult to compute for most random variables, $\mathrm{E}\left[\frac{1}{\Pi+1}\mathbb{1}_{\{\Pi \ge k\}}\right]$ can be expressed in compact form.

Lemma 1. For any integer k,

$$\mathrm{E}\left[\frac{1}{\Pi+1}\mathbb{1}_{\{\Pi \ge k\}}\right] = \frac{1}{m}\mathcal{P}\left[\Pi \ge k+1\right]$$

Proof. Immediate.

As shown in Lemma 2, this reciprocal moment is useful in finding tight bounds to the expectation of $\log \Pi$.

Lemma 2.

$$\log m - \log \frac{3}{2} + o(m^3 \mathrm{e}^{-m}) \leq \mathrm{E}\left[\log \Pi \mathbb{1}_{\{\Pi \geq 2\}}\right] \leq \mathcal{P}\left[\Pi \geq 2\right] \log m - \mathcal{P}\left[\Pi \geq 2\right] \log\left\{\mathcal{P}\left[\Pi \geq 2\right]\right\}$$

Proof. The upper bound is immediate from Jensen's inequality, while the lower bound is most easily seen from the inequality

$$\frac{1}{k} \leq \frac{3}{2}\frac{1}{1+k} \quad \forall k \geq 2$$

Over the range of integration, it follows that

$$\log \Pi \geq \log m - \log\left(\frac{3}{2}\frac{m}{1+\Pi}\right)$$

Taking the expectation over both sides for $\Pi \geq 2$, and applying Jensen's inequality to the second term, we have

$$\mathrm{E}\left[\log \Pi \mathbb{1}_{\{\Pi \geq 2\}}\right] \geq \mathcal{P}\left[\Pi \geq 2\right]\left\{\log m - \log\left[\frac{\frac{3}{2}\mathrm{E}\left[\frac{m}{\Pi+1}\mathbb{1}_{\{\Pi \geq 2\}}\right]}{\mathcal{P}\left[\Pi \geq 2\right]}\right]\right\}$$

Using Lemma 1 and the inequality $\log x \leq x - 1$, we get

$$\mathrm{E}\left[\log \Pi \mathbb{1}_{\{\Pi \geq 2\}}\right] \geq \log m - \log \frac{3}{2} - \left\{\mathrm{e}^{-m} + m\mathrm{e}^{-m}\right\}\log m + \frac{m^2}{2!}\mathrm{e}^{-m}$$

As seen from (3), Lemma 2 will be used to establish bounds on $\mathrm{E}\left[\log \Pi!\right]$. In addition, bounds on $\mathrm{E}\left[\Pi \log \Pi\right]$ will also be required. In Lemma 3, we use Lemma 1 to bound this expectation.

Lemma 3.

$$m \log m \leq \mathrm{E}\left[\Pi \log \Pi \mathbb{1}_{\{\Pi \geq 2\}}\right] \leq m \log m + 1$$

Proof. The left hand side follows from Jensen's inequality, while the right hand side follows from

$$\Pi \log \Pi = \Pi \log m + \Pi \log \left\{ 1 + \frac{\Pi - m}{m} \right\}$$

$$\leq \Pi \log m + \Pi \frac{\Pi - m}{m}$$

Taking the expectation of the right hand side for all values of Π produces the upper bound in the statement of the lemma.∎

As shown in (3) the tightness of the bounds of $\mathrm{E}\left[\log \Pi! \mathbb{1}_{\{\Pi \geq 2\}}\right]$ depends in part on the tightness of Stirling's bounds, which may be determined by $\mathrm{E}\left[\log\left(1 + \frac{1}{12\Pi - 1}\right) \mathbb{1}_{\{\Pi \geq 2\}}\right]$. In Lemma 4, we show that the bounds converge at least as fast as $\frac{1}{m}$.

Lemma 4.

$$\mathrm{E}\left[\log\left(1 + \frac{1}{12\Pi - 1}\right) \mathbb{1}_{\{\Pi \geq 2\}}\right] \leq \frac{1}{7m} \mathcal{P}\left[\Pi \geq 3\right]$$

Proof. For all $\Pi \geq 2$ we have

$$\frac{1}{12\Pi - 1} \leq \frac{1}{7(\Pi + 1)}$$

The lemma follows immediately from this inequality and Lemma 1.∎

From the Stirling bounds in (3) it is clear that bounds on $\mathrm{E}\left[\log \Pi!\right]$ follow from (3), as well as Lemmas 1 through 4. We combine and simplify these results in the following theorem.

Theorem 1. Let Π be a Poisson random variable with mean m. Given $\epsilon > 0$, there exists an m_ϵ so that for all $m > m_\epsilon$

$$\log \lfloor m \rceil! - \log \frac{3}{2} - \epsilon \leq \mathrm{E}\left[\log \Pi!\right] \leq \log \lfloor m \rceil! + 1 + \epsilon$$

where $\lfloor m \rceil!$ is Stirling's lower bound to $m!$.∎

Theorem 1 allows us to find an asymptotic form for the Poisson entropy. Recalling that Π is a Poisson random variable with mean m, the Poisson entropy may be written

$$H_P(m) \triangleq -\mathrm{E}\left[\log \frac{m^{\Pi} e^{-m}}{\Pi!}\right], \tag{4}$$

and our immediate aim is to show that

$$\lim_{m \to \infty} \frac{H_P(m)}{-\log \frac{m^m e^{-m}}{m!}} = 1.$$

The only quantity that is not known explicitly in (4) is $\mathrm{E}\left[\log \Pi!\right]$. As shown in Theorem 1, we can bound this expectation by quantities that differ by a constant as $m \to \infty$. More precisely, given $\epsilon > 0$, we may find an m_ϵ such that for all $m > m_\epsilon$,

$$\log \lfloor m \rceil! - \log \frac{3}{2} - \epsilon \leq \mathrm{E}\left[\log \Pi!\right] \leq \log \lfloor m \rceil! + 1 + \epsilon$$

where $\lfloor m \rceil!$ is Stirling's lower bound to $m!$. Using this result we tightly bound the Poisson entropy as the mean parameter m grows.

Theorem 2. Let Π be a Poisson random variable with mean m. Given $\epsilon > 0$, there exists an m_ϵ so that for all $m > m_\epsilon$ the Poisson entropy is bounded in the following way:

$$-\log \frac{m^m e^{-m}}{\lfloor m \rceil!} - \log \frac{3}{2} - \epsilon \leq H_P(m) \leq -\log \frac{m^m e^{-m}}{\lfloor m \rceil!} + 1 + \epsilon$$

where $\lfloor m \rceil!$ is Stirling's lower bound to $m!$. ■

This result is applicable to the PLPC channel, since the mutual information between the $\mathcal{X}_i$ and $\mathcal{Y}_i$ may be described in term of the Poisson entropy,

$$\mathcal{I}(\mathcal{X}_i; \mathcal{Y}_i) = H(\mathcal{Y}_i) - \mathrm{E}\left[H_P(\mathcal{X}_i + d)\right].$$

Conditioned on the input, the mean of $\mathcal{Y}_i$ is bounded from below by d, which grows arbitrarily large in the limit. This justifies the application of Theorem 2 to bound $\mathcal{I}(\mathcal{X}_i; \mathcal{Y}_i)$ for sufficiently large d.

Theorem 3. Given $\epsilon > 0$ there exists a d_ϵ such that for all average noise energies $d > d_\epsilon$ and all distributions on $\mathcal{X}$ such that $\mathcal{X} \geq 0$ *a.s.*,

$$-1 - \epsilon \leq \mathcal{I}(\mathcal{X}_i; \mathcal{Y}_i) - H(\mathcal{Y}_i) - \mathrm{E}\left[\log \frac{(\mathcal{X}_i + d)^{(\mathcal{X}_i + d)} e^{-(\mathcal{X}_i + d)}}{\lfloor (\mathcal{X}_i + d) \rceil!}\right] \leq \log \frac{3}{2} + \epsilon.$$

■

By substituting the definition of Stirling's lower bound, these bounds take a form which is more amenable to bounding

$$-1 - \epsilon \leq \mathcal{I}(\mathcal{X}_i; \mathcal{Y}_i) - H(\mathcal{Y}_i) + \mathrm{E}\left[\frac{1}{2}\log[2\pi(\mathcal{X}_i + d)]\right] \leq \log\frac{3}{2} + \epsilon. \tag{5}$$

We shall develop bounds on the capacity of the PLPC by bounding the output entropy $H(\mathcal{Y}_i)$ in (5). As in [5] we achieve a lower bound on $\mathcal{I}(\mathcal{X}_i; \mathcal{Y}_i)$ by assuming an exponential distribution on the input.

Theorem 4. Let $\bar{\mathcal{X}}, d \to \infty$ proportionally. Given $\epsilon > 0$ there exists an $\bar{\mathcal{X}}_\epsilon$ such that for all $\bar{\mathcal{X}} > \bar{\mathcal{X}}_\epsilon$ the capacity of the PLPC is lower bound by

$$\mathcal{C} \geq \frac{1}{2}\log\frac{\bar{\mathcal{X}}}{2\pi} - \frac{1}{2}\log\left(1 + \frac{1}{SNR}\right) - \epsilon$$

where $SNR = \bar{\mathcal{X}}/d$.

Proof. When the input to the channel is exponential with mean $\bar{\mathcal{X}}$, it can be shown that the output distribution has the form

$$\begin{aligned}\mathcal{P}\left[\mathcal{Y} = y\right] &= (1-\rho)\rho^y e^{\frac{1}{SNR}}\mathcal{P}\left[\Pi \leq y\right] \\ &\leq (1-\rho)\rho^y e^{\frac{1}{SNR}}\end{aligned} \tag{6}$$

where $\rho \overset{\Delta}{=} \frac{\bar{\mathcal{X}}}{1+\bar{\mathcal{X}}}$, and Π is a Poisson random variable with mean $m = d + \frac{1}{SNR}$. A lower bound on the output entropy follows from (6),

$$H(\mathcal{Y}_i) \geq \log[1 + \bar{\mathcal{X}}] + (d + \bar{\mathcal{X}})\log\left[1 + \frac{1}{\bar{\mathcal{X}}}\right] - \frac{d}{\bar{\mathcal{X}}} \tag{7}$$

The theorem follows by substituting (7) in (5), and by applying Jensen's inequality to the term $\mathrm{E}\left[\frac{1}{2}\log[2\pi(\mathcal{X}_i + d)]\right]$ in (5). ∎

We achieve the upper bound to the PLPC capacity by lower bounding the term $\mathrm{E}\left[\frac{1}{2}\log[2\pi(\mathcal{X}_i + d)]\right]$ in (5), and by assuming that there is some distribution on $\mathcal{X}_i$ such that $\mathcal{Y}_i$ is geometric with mean $\bar{\mathcal{X}} + d$. This follows from the fact that the geometric random variable maximizes the entropy functional over all distributions on the integers subject to a constrained mean. While it can be shown that there is no distribution on $\mathcal{X}_i$ such that $\mathcal{Y}_i$ is geometrically distributed for $d > 0$, it is clear that the entropy of a geometric random variable with mean $\bar{\mathcal{X}} + d$ is an upper bound to $H(\mathcal{Y}_i)$.

Theorem 5. Let $d, \bar{\mathcal{X}} \to \infty$ proportionally. Given $\epsilon > 0$ there exists a $\bar{\mathcal{X}}_\epsilon$ such that for all $\bar{\mathcal{X}} > \bar{\mathcal{X}}_\epsilon$ the capacity of the PLPC is upper bounded by

$$\mathcal{C} \leq \frac{1}{2}\log\frac{\bar{\mathcal{X}}}{2\pi} + \log\left(\sqrt{SNR}(1+\frac{1}{\bar{\mathcal{X}}_\epsilon}) + \frac{1}{\sqrt{SNR}}\right) + 1 + \log\frac{3}{2} + \epsilon. \qquad (8)$$

Proof. The upper bound may be achieved by replacing the expectation in (5) by the minimum value of the integrand, $\frac{1}{2}\log 2\pi d$, and by using the entropy of the geometric random variable in place of $H(\mathcal{Y}_i)$. In this way we find

$$H(\mathcal{Y}_i) \leq \log(1+\bar{\mathcal{X}}+d) + (\bar{\mathcal{X}}+d)\log\left(1+\frac{1}{\bar{\mathcal{X}}+d}\right)$$

and the bound follows immediately.■

4. *References*

[1] Y.M. Kabanov, *"The Capacity of a Channel of the Poisson Type", Theory of Probability and Applications,* **Vol. 23,** *January, 1978, pp. 143-147*

[2] A.A. Lazar, *"On the Capacity of the Poisson Type Channel", Proceedings of the Fourteenth Annual Conference on Information Sciences and Systems, Princeton, NJ, March 26-28, 1980, pp.285-289*

[3] M.H.A. Davis, *"Capacity and Cutoff Rate for Poisson-Type Channels", IEEE Transactions on Information Theory,* **Vol. IT-26, No. 6,** *November, 1980, pp. 710-715*

[4] A.D. Wyner, *"Capacity and Error Exponent for the Direct Detection Photon Channel-Part I", IEEE Transactions on Information Theory,* **Vol. IT-34, No. 6,** *November, 1988, pp. 1449-1461*

[5] J.P. Gordon, *"Quantum Effects in Communications Systems", Proceedings of the IRE,* **Vol. 50,** *September, 1962, pp. 1898-1908*

[6] E. Hisdal, *"Information in a Photon Beam vs. Modulation-Level Spacing," Journal of the Optical Society of America,* **Vol. 61, No.3,** *March 1971, pp.328-332*

[7] S. Shamai (Shitz), *"The Capacity of a Pulse Amplitude Modulated Direct Detection Photon Channel," preprint*

[8] J.G. Smith, *"The Information Capacity of Amplitude- and Variance-Constrained Scalar Gaussian Channels," Information and Control,* **Vol. 18,** *pp.203-219, 1971*

[9] I. Bar-David, *"Error Probabilities in Random-Phase Binary Optical Signalling", IEEE Transactions on Information Theory, May, 1974*

RESTRICTED TWO-WAY CHANNELS WITH CORRELATED SOURCES

Masoud Salehi
CDSP, Department of Electrical and Computer Engineering
Northeastern University, Boston MA 02115

Abstract : *We will investigate the problem of transmitting two correlated sources via a restricted two-way channel. It is shown that by employing correlation between the sources it is possible to operate beyond the capacity region for the case of independent inputs. A coding theorem is proved and an achievable rate region is derived. It is shown that if the correlation between sources matches the statistical properties of the channel, this achievable rate region coincides with the Shannon's outer bound. Finally we will give an expression, in the form of a limit, for the optimal matching conditions in this case.*

Two-way channel (TWC) was introduced by Shannon[1] as a model for communication between two parties. In the same work Shannon derived inner and outer bounds for C_{TWC}, the capacity region of the TWC. The inner bound $\mathcal{G}_i$ is defined by

$$\mathcal{G}_i = \mathcal{CO}\{\cup_{p(x_1,x_2)=p(x_1)p(x_2)}\{(R_1,R_2) : 0 \le R_1 \le I(X_1;Y_1|X_2), 0 \le R_2 \le I(X_2;Y_2|X_1)\}\}$$

where $\mathcal{CO}$ denotes the convex hull operation. The outer bound $\mathcal{G}_o$ is

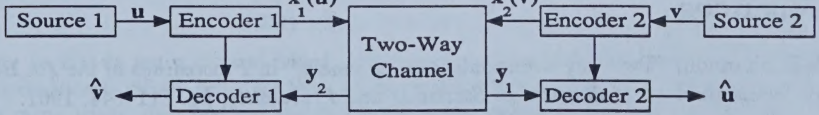
$$\mathcal{G}_o = \cup_{p(x_1,x_2)}\{(R_1,R_2) : 0 \le R_1 \le I(X_1;Y_1|X_2), 0 \le R_2 \le I(X_2;Y_2|X_1)\}$$

(X_1 and X_2 denote the input random variables and Y_1 and Y_2 denote the output variables). Shannon proved that $\mathcal{G}_i \le C_{\mathrm{TWC}} \le \mathcal{G}_o$ and showed that in general $\mathcal{G}_i \subset \mathcal{G}_o$.

The restricted two-way channel (RTWC) is defined as a TWC for which the encoding functions depend on the messages only, i.e. in a RTWC there is no feedback available from the outputs to the encoders. Shannon showed that for the RTWC the capacity region coincides with the inner bound.

In this work we will investigate the problem of transmission of correlated sources via the RTWC. We will prove a coding theorem for this case. It will be shown that by using the dependence between the sources we can achieve higher transmission rates than the case of independent messages. Also it is shown that if the sources are not independent separate source and channel coding is not optimal.

The communication situation we are going to investigate is depicted below: The source



are generated by independent drawings of a pair of random variables U and V distributed according to some PMF $p(u,v)$, the encoding schemes are defined by $\mathbf{x}_1(\mathbf{u}) : \mathcal{U}^n \to \mathcal{X}_1^n$ and $\mathbf{x}_2(\mathbf{v}) : \mathcal{V}^n \to \mathcal{X}_2^n$ and the decoding functions are $\hat{\mathbf{u}} : \mathcal{V}^n \times \mathcal{Y}_1^n \to \mathcal{U}^n$ and $\hat{\mathbf{v}} : \mathcal{U}^n \times \mathcal{Y}_2^n \to \mathcal{V}^n$. Error probabilities are defined as usual. For this communication situation we can prove the following theorem.

THEOREM 1: Let the source pair $(\mathbf{U}, \mathbf{V})$ be generated by independent drawings of U and V distributed according to $p(u, v)$ and let the restricted two-way channel be denoted by $p(y_1, y_2|x_1, x_2)$. Define the region $\mathcal{R}_{cs}$ by

$$\mathcal{R}_{cs} = \mathcal{CO}\{\cup\{(R_1, R_2) : 0 \leq R_1 \leq I(X_1; Y_1|X_2, V), 0 \leq R_2 \leq I(X_2; Y_2|X_1, U)\}\}$$

where the union is over all choices of $p(x_1|u)$ and $p(x_2|v)$ such that $p(u, v, x_1, x_2, y_1, y_2) = p(u, v)p(x_1|u)p(x_2|v)p(y_1, y_2|x_1, x_2)$. Then reliable communication of the source over the channel is possible if the condition $(H(U|V), H(V|U)) \in \mathcal{R}_{cs}$ is satisfied.

Remark : By introducing a time-sharing random variable T, we can redefine $\mathcal{R}_{cs}$ as

$$\mathcal{R}_{cs} = \cup\{(R_1, R_2) : 0 \leq R_1 \leq I(X_1; Y_1|X_2, V, T), 0 \leq R_2 \leq I(X_2; Y_2|X_1, U, T)\}$$

where the union is over all choices of $p(t)$, $p(x_1|u, t)$ and $p(x_2|v, t)$ such that

$$p(t, u, v, x_1, x_2, y_1, y_2) = p(t)p(u, v)p(x_1|u, t)p(x_2|v, t)p(y_1, y_2|x_1, x_2)$$

and the cardinality of T is 2.

Example : Let U and V be binary correlated with $p(0, 0) = 0, p(1, 0) = p(0, 1) = 0.275$ and $p(1, 1) = 0.45$ and let the channel be a binary multiplying channel with $Y_1 = Y_2 = X_1 \times X_2$, then by letting $x_1 = u$ and $x_2 = v$ we can achieve the point $(0.69424, 0.69424)$ which lies on the Shannon's outer bound[1].

PROOF : The proof of the theorem is based on random coding employing correlation preserving technique, for details see[2].

Theorem 1 only gives the sufficient conditions for matching of the source to the channel, the following theorem gives the necessary and sufficient conditions although not in the single letter form.

THEOREM 2: Let the source and the channel be defined as in theorem 1, then reliable transmission of the source over the channel is possible if and only if $(H(U|V), H(V|U)) \in \mathcal{C}$ where $\mathcal{C} = \cup_{k=1}^{\infty} \mathcal{C}_k$ and $\mathcal{C}_k$ is defined by

$$\mathcal{C}_k = \{(R_1, R_2) : 0 \leq R_1 \leq \frac{1}{k} I(\mathbf{X}_1^k; \mathbf{Y}_1^k|\mathbf{X}_2^k, \mathbf{V}^k), 0 \leq R_2 \leq \frac{1}{k} I(\mathbf{X}_2^k; \mathbf{Y}_2^k|\mathbf{X}_1^k, \mathbf{U}^k)\}$$

where the union is over all choices of $p(\mathbf{x}_1^k|\mathbf{u}^k)$ and $p(\mathbf{x}_2^k|\mathbf{v}^k)$ such that

$$p(\mathbf{u}^k, \mathbf{v}^k, \mathbf{x}_1^k, \mathbf{x}_2^k, \mathbf{y}_1^k, \mathbf{y}_2^k) = [\Pi_{i=1}^k p(u_i, v_i)] p(\mathbf{x}_1^k|\mathbf{u}^k) p(\mathbf{x}_2^k|\mathbf{v}^k) [\Pi_{i=1}^k p(y_{1i}, y_{2i}|x_{1i}, x_{2i})]$$

Remark : The region $\mathcal{C}$ as defined above is already convex therefore no time-sharing is necessary.

References

[1] C. E. Shannon, "Two-way communication channels," in *Proceedings of the 4th Berkeley Sumposium on Mathematical Statistics and Probability*, pp. 611–644, 1961.

[2] T. M. Cover, A. ElGamal, and M. Salehi, "Multiple-access channels with arbitrarily correlated sources," *IEEE Transactions on Information Theory*, vol. IT-26, pp. 648–657, November 1980.

EXACT ASYMPTOTIC STATISTICS FOR SIGMA-DELTA QUANTIZATION NOISE

David F. Delchamps
School of Electrical Engineering
Cornell University
Phillips Hall
Ithaca, NY 14853

ABSTRACT

Using a fundamental result from ergodic theory, we show that generic T-periodic inputs to a single-stage oversampled sigma-delta modulator lead to quantization error sequences with well-defined asymptotic statistics of all orders. For such inputs, the values of the asymptotic statistics are independent of the initialization of the delay element in the modulator and of the initial phase on the input. Our results generalize some formulas derived recently by R. M. Gray for the constant input case.

1. Introduction

For the purposes of this paper, an *analog-to-digital converter* (or ADC) is a dynamical system that accepts as its input a real-valued continuous-time signal $t \to x(t)$, $t \in \mathrm{R}$, and generates as its output a discrete-time signal $n \to \hat{x}_n$, $n \in \mathrm{Z}$, taking values in a finite or countable set J. One can think of $n \to \hat{x}_n$ as a discrete-valued signal that contains certain information about the input signal $t \to x(t)$; the manner in which this information is "encoded" in $n \to \hat{x}_n$ depends upon how the analog-to-digital converter operates.

Perhaps the simplest example of an analog-to-digital converter is the *memoryless uniform ADC,* which can be described as follows. Let M be an even positive integer and $\Delta > 0$ a real number. Let q be the *M-level uniform quantizer with sensitivity* Δ defined by

$$q(x) = (j+\tfrac{1}{2})\Delta, \qquad x \in [j\Delta, (j+1)\Delta), \qquad |j| < \tfrac{1}{2}M \ ,$$

and $q(x) = \tfrac{1}{2}jM\Delta$ when $x \in [j\Delta, (j+1)\Delta)$ and $|j| \geq \tfrac{1}{2}M$. The quantizer q is a piecewise constant mapping that takes values in the finite set $\{j\Delta : |j| \leq \tfrac{1}{2}M\}$. If $\tau_o > 0$ is given, the M-level memoryless uniform ADC with sensitivity Δ and sampling interval τ_o takes an input signal $t \to x(t)$, $t \in \mathrm{R}$, and generates the sequence $\{\hat{x}_n = q(x(n\tau_o)) : n \in \mathrm{Z}\}$. For each n, $\hat{x}_n$ is a uniformly quantized version of the value of $t \to x(t)$ at the "nth sampling instant" $n\tau_o$. One usually assumes that $M\Delta$ is large enough so that $|x(t)| \leq \tfrac{1}{2}M\Delta$ for all t; in other words, the value of $x(t)$ never strays into the "overload region" of the quantizer q. Another common requirement is that τ_o be at least as small as the Nyquist sampling interval when the input $t \to x(t)$ is bandlimited. The "dynamics" of the memoryless uniform ADC are rather trivial; the way $n \to \hat{x}_n$ stores information about $t \to x(t)$ is elementary, as well. Indeed, if we construct the continuous-time signal $t \to \bar{x}(t)$ by passing $n \to \hat{x}_n$ through a *zero-order-hold,* i.e.

$$\bar{x}(t) = \hat{x}_n \ , \qquad t \in [n\tau_o, (n+1)\tau_o) \ ,$$

then $t \to \bar{x}(t)$ is just a discrete-valued "staircase approximation" of $t \to x(t)$. The accuracy of approximation is clearly better for large M and small Δ; that is, if q is a very sensitive quantizer with many levels, then one can obtain a good approximation of $t \to x(t)$, at least at sampling instants, by processing $n \to \hat{x}_n$.

The significant practical problems associated with building a highly sensitive uniform quantizer with many levels are among the motivations for using *oversampled analog-to-digital converters,* of which sigma-delta modulators are good examples. To understand in general terms how an oversampled ADC works, suppose our objective is to obtain a discrete-valued staircase approximation to an input signal $t \to x(t)$, and we want this approximation to be constant over intervals of the form $[n\tau_o, (n+1)\tau_o)$. We sample $t \rightarrow x(t)$ every $\tau = \tau_o/N$ seconds, where N is a positive integer, then generate a discrete-valued approximation $k \rightarrow \tilde{x}_k$ to the sequence $k \rightarrow x(k\tau)$, $k \in$ Z. Next, we *decimate* the $\{\tilde{x}_k\}$ sequence to get $\hat{x}_n = \tilde{x}_{nN}$, $n \in$ Z, and form the staircase approximation $t \rightarrow \bar{x}(t)$ from $\{\hat{x}_n\}$ using a τ_o-zero-order-hold just as in the case of the memoryless uniform ADC. An important advantage of oversampled ADCs is that generating a good staircase approximation of $t \rightarrow x(t)$ does not require that the *quantizers* the converter employs be extremely sensitive or have extraordinarily many levels. Speaking very loosely, oversampled ADCs make clever use of an abundance of rough approximations to generate a smaller number of good approximations. We don't have the space here to penetrate much more deeply the motivations for oversampled ADCs, and refer the reader instead to [4], [5], and the references therein.

Depicted in Figure 1 is the block diagram for a standard sigma-delta modulator. In the notation of the diagram, which is precisely that used by Gray in [5], x_k represents the value at time k of the discrete-time signal one obtains by oversampling the input signal $t \rightarrow x(t)$ every τ seconds. The quantizer q is a uniform M-level quantizer with sensitivity Δ; in most applications, q is a *binary* quantizer with $M = 2$ and $\Delta = 1$. The H-block represents a linear time-invariant filter, which more often than not is simply a finite impulse response averaging filter, so that in the notation of Figure 1 we have

$$\tilde{x}_k = \frac{1}{N} \sum_{l = k-N}^{k-1} q(u_l), \quad k \in \mathrm{Z}.$$

One decimates the signal $k \rightarrow \tilde{x}_k$ to obtain $n \rightarrow \hat{x}_n$ via $\hat{x}_n = \tilde{x}_{nN}$, $n \in$ Z. The presence of the delay block makes the system genuinely dynamical.

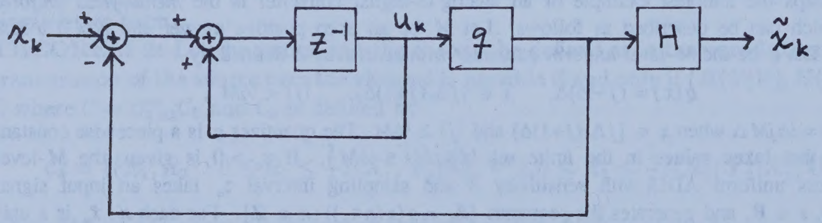


Figure 1

To understand roughly how the sigma-delta modulator works, it helps to think of the output $q(u_k)$ of the quantizer as u_k plus an "error term" $\varepsilon_k = q(u_k) - u_k$. If q were absent from the picture, we would have $u_k = x_{k-1}$ for every k, and it would be fairly clear why $\hat{x}_n$ approximated $x(t)$ for $t \in [n\tau_o, (n+1)\tau_o)$, particularly if $t \rightarrow x(t)$ varied only slowly. A moment's reflection about the effect of q explains at least partly the engineering rationale for oversampling $t \rightarrow x(t)$ and averaging $k \rightarrow x_k$ through H. If we assume that the error sequence $k \rightarrow \varepsilon(k)$ is well-approximated over the long haul by white noise distributed uniformly on $[-\frac{1}{2}\Delta, \frac{1}{2}\Delta]$, then for large "oversampling ratio" N we might expect the averaging filter H to mitigate the deleterious effects of the quantization error. It is therefore important to try to answer the following question: under what circumstances is the quantization "noise" sequence $k \rightarrow \varepsilon(k)$ (at least approximately) "asymptotically uniform and white?" We shall find in what follows that for "almost all" periodic inputs $t \rightarrow x(t)$, the sequence $k \rightarrow \varepsilon(k)$, while not "white" by any means, is asymptotically uniform in a strong sense.

2. A Generalization of Gray's Formulas for Bandlimited Inputs

The question at the end of the preceding section has a long and colorful history, and has arisen in contexts far removed from oversampled analog-to-digital conversion. An early paper describing some conditions guaranteeing uniform and white quantization errors is [1]; during the forty-odd years since [1] appeared, most researchers have shied away from trying to answer the question and have focused instead on designing systems under the assumption that one could model error sequences arising from uniform quantizers as uniform white noises. Over the last few years, however, several investigators, most notably R. M. Gray and his colleagues (see [4], [5], and references therein), have made significant progress towards obtaining a precise understanding of the asymptotics of quantization error sequences arising in a variety of uniform quantizer applications including oversampled sigma-delta modulation. Related work employing an entirely different set of techniques in a control systems context appears in [2].

A reasonable way to open any investigation of sigma-delta quantization error sequences is to assume that the input $t \longrightarrow x(t)$ is constant and attempt to obtain some insight about the asymptotic first- and second-order statistics of the error sequence $k \longrightarrow \varepsilon(k)$. In [5], after intrepidly traversing a daunting analytical jungle rife with Bessel functions and equidistribution results for arithmetic sequences, Gray arrives at the following remarkable conclusions: for the sigma-delta modulator of Figure 1, if $u_0 = 0$ and $\{e_k\}$ denotes the normalized quantization error sequence $\{\Delta^{-1}\varepsilon_k\}$, then for "most" constant inputs $x_k \equiv \bar{x}$, $k \geq 0$, one has

$$\bar{e} \overset{\Delta}{=} \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} e_l = 0\,, \tag{GF1}$$

and

$$R_e(k) \overset{\Delta}{=} \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} e(l)e(l+k) = \frac{1}{12} - \frac{1}{2} <k\beta>_1 (1 - <k\beta>_1)\,, \tag{GF2}$$

where $\beta = \frac{1}{2} + \Delta^{-1}\bar{x}$ and $<k\beta>_1$ denotes the fractional part (mod 1) of $k\beta$. The proofs of Gray's Formulas (GF1) and (GF2) require only that the input's constant value $\bar{x}$ be an irrational multiple of Δ that lies in the "no overload range" $(-\frac{1}{2}(M-1)\Delta, \frac{1}{2}(M-1)\Delta)$ of the quantizer q.

Demonstrating the mere existence of limits such as those appearing in (GF1) and (GF2) is rarely an easy exercise, particularly if the signal $k \longrightarrow e(k)$ comes out of a system with a discontinuous nonlinearity buried in a feedback loop; for a discussion of the kinds of complications one might encounter, see [2]. It is fortuitous that the limiting autocorrelation $R_e(k)$ for the error sequence has such an economical and aesthetically appealing form. It follows immediately from (GF2), for example, that the time-average of $e^2(k)$ is 1/12, which would be true if $\{e(k)\}$ were a sequence of independent uniform random variables. It is also evident from (GF2) that $R_e(k)$ is not in general zero for nonzero k, which means that the sequence $\{e(k)\}$ is not always "white" by any means. We should note that Kieffer (see, for example, [6]), by using some powerful techniques from symbolic dynamics, has extended (GF1) and (GF2) to cover more general ADCs driven by constant inputs, and that Gray and Kieffer (see [5] and [6]) have both investigated the implications of (GF1) and (GF2) and their generalizations with regard to asymptotic *spectral properties* of the normalized quantization error sequence $k \longrightarrow e(k)$. Moreover, Gray investigates in [5] the asymptotic statistics of $\{e(k) : k \geq 0\}$ for the sigma-delta modulator of Figure 1 driven by certain sinusoids and random inputs, and for multi-stage sigma-delta modulators. In none of these cases, however, do the computations terminate in formulas as elegant as (GF1) and (GF2).

It is the purpose of the present paper to prove a result that generalizes Gray's Formulas significantly; our result not only allows time-varying inputs $t \longrightarrow x(t)$, but enables us to make strong statements about all the higher-order asymptotic statistics of the error sequence $k \longrightarrow \varepsilon(k)$. We shall be using techniques entirely different from the ones employed by Gray in deriving (GF1) and (GF2). The key ingredients are some fundamental results from modern ergodic theory, particularly as it applies to continuous maps of circles and tori. What makes these results applicable is the *dynamical* nature of the sigma-delta modulator in Figure 1, which we exploit by adopting a viewpoint that is in some sense "90° out of phase" with the one most commonly assumed by digital communications theorists. Instead of fixing u_0, the initial state of the delay, and subsequently regarding the sigma-delta modulator as an input-output device, we fix instead the input function $t \to x(t)$, and investigate the (possibly nonautonomous) one-dimensional dynamical system that results. The state of this dynamical system at time k is the state u_k of the integrator. The system's dynamics are nonlinear because of the presence of the quantizer q.

To demonstrate our plan of attack, it is useful to to examine first the situation where the input function $t \to x(t)$ is identically equal to a constant value $\bar{x}$. It is not hard to demonstrate (see [5]) that if neither $|u_0|$ nor $|\bar{x}|$ exceeds $b = \frac{1}{2}(M-1)\Delta$, then $|u_k| \le b$ for every $k \ge 0$. In other words, if both the initial state of the delay and the value of the input are in the "no overload range" of the quantizer q, then the integrator state stays in the no overload range forever. As a consequence, the quantization error ε_k always lies in the interval $[-\frac{1}{2}\Delta, \frac{1}{2}\Delta]$, so that $e_k = \Delta^{-1}\varepsilon_k \in [-\frac{1}{2}, \frac{1}{2}]$ A modicum of algebra (see [5] for details) leads to the following recursion for e_k:

$$e_{k+1} = \frac{1}{2} - < \frac{x}{\Delta} - e_k >_1 ,$$

where $< \; >_1$, as before, denotes fractional part mod 1. It turns out to the easier to regard $y_k = \frac{1}{2} - e_k$ as the state at time k of the system, so we focus on the equation

$$y_{k+1} = G(y_k) = < y_k + \frac{1}{2} + \frac{\bar{x}}{\Delta} >_1 .$$

The piecewise linear mapping G takes the unit interval $[0, 1]$ into itself. Although G is discontinuous as a map of the interval, it lifts to a mapping $\tilde{G}$ from the unit circle into itself that *is* continuous; in fact, $\tilde{G}$ simply rotates points on the unit circle counterclockwise through an angle $2\pi\beta = 2\pi(\frac{1}{2} + \Delta^{-1}\bar{x})$. Much is known about the asymptotics of rotations of the unit circle such as $\tilde{G}$; most ergodic theory texts (see, for example, [7]) contain a proof that if β is irrational, then $\tilde{G}$ is *strictly ergodic* in the sense that (normalized) Lebesgue measure on the circle is the only Borel (probability) measure preserved by $\tilde{G}$, and that for any Borel measurable function $\tilde{f}$ on the unit circle, the sequence

$$\{\frac{1}{L}\sum_{l=0}^{L-1} \tilde{f}(\tilde{G}^l(e^{i2\pi y_0})) : 1 \le L < \infty\} \qquad (*)$$

converges for Lebesgue almost every $y_0 \in [0, 1]$ to

$$\int_0^1 \tilde{f}(e^{i2\pi y})dy ;$$

here, $\tilde{G}^k$ denotes the k-fold composition of $\tilde{G}$ with itself. It turns out that when $\tilde{f}$ is continuous, the sequence (*) converges *uniformly* in y_0 and therefore converges for *every* y_0 (see [3]).

Applying this result to our problem is straightforward. If $f : [0, 1] \to \mathbf{R}$ is Borel measurable, it lifts (nonuniquely) to a Borel measurable function $\tilde{f}$ on the unit circle; every such lifting satisfies

$$\tilde{f}(e^{i2\pi y}) = f(y)$$

for every $y \in [0, 1]$, and therefore

$$\lim_{L \to \infty} \frac{1}{L}\sum_{l=0}^{L-1} f(G^l(y_0)) = \lim_{L \to \infty} \frac{1}{L}\sum_{l=0}^{L-1} \tilde{f}(\tilde{G}^l(e^{i2\pi y_0}))$$

whenever the limits exist. Observe that any lifting $\tilde{f}$ of f will be continuous if f is continuous on $[0, 1]$ and satisfies $f(1) = f(0)$. Hence, if β is irrational and $f : [0, 1] \to \mathbf{R}$ is continuous and satisfies $f(0) = f(1)$, we have

$$\lim_{L \to \infty} \frac{1}{L}\sum_{l=0}^{L-1} f(G^l(y_0)) = \int_0^1 f(y)dy$$

for every $y_0 \in [0, 1]$. In other words, for every initial state y_0, the resulting sequence of system states $\{y_l = G^l(y_0)\}$ satisfies

$$\lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} f(y_l) = \int_0^1 f(y)dy \qquad (**)$$

for every continuous function $f : [0, 1] \to \mathbf{R}$ satisfying $f(0) = f(1)$.

Using standard techniques from integration theory, we can extend (**) to cover various functions $f : [0, 1] \to \mathbf{R}$ that aren't continuous; in fact, (**) holds for any f for which we can find, given $\delta > 0$, two continuous functions f_* and f_* that bound f pointwise from below and above in the sense that

$$f_*(y) \leq f(y) \leq f^*(y)$$

for every $y \in [0, 1]$, and at the same time satisfy

$$\int_0^1 (f^*(y) - f_*(y))dy < \delta .$$

The family of f's that can be so approximated includes all the piecewise continuous functions on $[0, 1]$ that have only finitely many jump discontinuities, even those that don't satisfy $f(0) = f(1)$. Note, however, that the (measurable) function that equals 1 on the rationals and zero on the irrationals admits no such approximation. Taking f in (**) as the indicator function 1_V of an interval $V \subset [0, 1]$ yields

$$\lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} 1_V(y_l) = \lambda(V) ,$$

where λ denotes Lebesgue measure; we conclude that *regardless of the initial condition* y_0, the "long-term fraction of the time" that the sequence $k \to y_k$ spends in V is equal to the length of V, which means that the sequence $\{y_k\}$ possesses a certain kind of *asymptotic uniformity* over the interval $[0, 1]$. Since $e_l = \frac{1}{2} - y_l$, the same sort of assertion holds for the sequence $\{e_k\}$ regarding its asymptotic distribution over the interval $[-\frac{1}{2}, \frac{1}{2}]$.

Setting $f(y) = y$ in (**) yields

$$\lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} (y_l) = \int_0^1 y dy = \frac{1}{2} ,$$

so that the time average of the sequence $\{y_k\}$ is well-defined and equal to $\frac{1}{2}$, once again independently of the initial condition y_0; (GF1) follows immediately from $e_k \equiv \frac{1}{2} - y_k$. If we take f in (**) to be the (piecewise continuous) function defined by

$$f(y) = yG^k(y) , \quad y \in [0, 1] ,$$

then we get

$$\lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} (y_l y_{l+k}) = \int_0^1 yG^k(y)dy .$$

It is easy to see that $G^k(y) = <y + k\beta>_1$, and this identity makes evaluating the last integral straightforward. Writing e_k in terms of y_k gives (GF1).

In the fashion we've just described, a simple application of ergodic theory to the case where $x(t) \equiv \bar{x}$, with $\beta = \frac{1}{2} + \Delta^{-1}\bar{x}$ irrational, leads not only to Gray's formulas (GF1) and (GF2) but also to a statement about the *strong asymptotic uniformity in* $[0, 1]$ of the normalized quantization error sequence $\{e_k\}$. Moreover, the ergodic theoretic approach shows that (GF1) and (GF2), along with the asymptotic uniformity results, hold *independently of the initial condition* y_0. (N.B.: in [5], Gray asserts (correctly) without proof that (GF1) and (GF2) hold independently of y_0.) We can view our result probabilistically, as well: under the assumption that β is irrational (which is equivalent to the assumption that the constant input value $\bar{x}$ is not a rational multiple of Δ), the sequence $\{e_k\}$ behaves asymptotically like a sample path of a zero-mean stationary uniform random process with autocorrelation function given by (GF2). It is clear how one can use the technique just outlined to derive asymptotic statistics *of all orders* for the sequence $\{e_k\}$, at least in the case of a constant input to the sigma-delta modulator. If one wants Jth-order statistics then one applies (**) with

$$f(y) = yG^{k_1}(y)G^{k_2}(y) \ldots G^{k_{J-1}}(y)$$

for all the different integer $(J-1)$-tuples $(k_1, \ldots, k_{J-1})$.

We turn now to the case where the input $t \to x(t)$ is not constant. The foregoing computation goes through up to a point; the revised recursions for the $\{e_k\}$ and $\{y_k\}$ sequences are

$$e_{k+1} = \frac{1}{2} - <\frac{x_k}{\Delta} - e_k>_1$$

and

$$y_{k+1} = <y_k + \frac{1}{2} + \frac{x_k}{\Delta}>_1 ,$$

where $x_k = x(t_0 + k\tau)$, with $t_0 \in \mathbf{R}$ some arbitrary "initial phase." These recursions are time-varying; how, one might ask, does ergodic theory apply?

The trick we employ is one of the oldest in the system theorist's book, and isn't often very helpful; fortunately, in this case, it is practically made to order. The idea is to "turn time into a state variable," and thus make the one-dimensional time-varying recursion for y_k into a two-dimensional time-invariant one. Accordingly, we restrict our attention to *T-periodic inputs* $t \to x(t)$, and regard the system's dynamics as being governed by the following pair of recursions:

$$\begin{aligned} y_{k+1} &= <y_k + \frac{1}{2} + \frac{x(t_k)}{\Delta}>_1 \\ t_{k+1} &= <t_k + \tau>_T , \end{aligned}$$

where $<\ >_T$ represents fractional part mod T. These equations define a mapping $\tilde{G}$ from the two-dimensional torus to itself; if we set $z_1(k) = e^{i2\pi y_k}$ and $z_2(k) = e^{i\omega t_k}$ (where $\omega = 2\pi T^{-1}$), then $\tilde{G}$ takes the form

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \overset{\tilde{G}}{\to} \begin{bmatrix} g(z_2)z_1 \\ e^{i\omega\tau}z_2 \end{bmatrix} .$$

The function g will be continuous if $t \to x(t)$, in addition to being T-periodic, is piecewise continuous with jumps whose sizes are integer multiples of Δ.

In [3], Furstenberg studies mappings such as $\tilde{G}$ in great detail. The principal result of [3] gives a condition under which $\tilde{G}$ is *uniquely ergodic*, which means as before that Lebesgue measure is the only Borel probability measure on the torus invariant under $\tilde{G}$ and that "the time average" of any continuous function evaluated along *any* (not just "almost any") trajectory of the mapping $\tilde{G}$ converges to the "space average" of the function with respect to Lebesgue measure on the torus. Furstenberg's Theorem, reshaped to fit our notation and context, reads as follows: if the function g arising from x is continuous, then $\tilde{G}$ is uniquely ergodic if and only if there do not exist an integer $m \neq 0$ and a Borel measurable function $z_2 \to R(z_2)$ satisfying $|R(z_2)| \equiv 1$ along with

$$R(e^{i\omega(t+\tau)}) = R(e^{i\omega t})e^{i2\pi m(\frac{1}{2}+\Delta^{-1}x(t))} \tag{F}$$

for every $t \in [0, 1]$. We concentrate for the remainder of the paper on obtaining conditions involving the input $t \to x(t)$ and the parameters of the sigma-delta modulator in Figure 1 guaranteeing that Furstenberg's condition (F) does not hold for any function R and nonzero integer m. The sufficient conditions we derive are mercifully weak, and we are therefore able to conclude that a "generic" T-periodic input $t \to x(t)$ leads to a uniquely ergodic $\tilde{G}$. Unique ergodicity of $\tilde{G}$

ensures in turn that for every initial condition (t_0, y_0) the resulting $\{y_k\}$ and $\{e_k\}$ sequences are strongly asymptotically uniform, and that the following equation holds for "most" functions $f : [0,1] \times [0,T] \to \mathrm{R}$ independently of initial conditions:

$$\lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} f(y_l, t_l) = \frac{1}{T} \int_0^T \int_0^1 f(y,t)\, dy\, dt \; . \tag{***}$$

As with (**) in the case of a constant input, instantiations of (***) amount to weighty assertions about the asymptotic statistics of the $\{y_k\}$ and $\{e_k\}$ sequences. Two of these assertions are direct generalizations of Gray's Formulas.

The following definition introduces the condition that provides the key to the ensuing analysis.

2.1 Definition: Let $x : \mathrm{R} \to \mathrm{R}$ be piecewise continuous and T-periodic; assume that the mean value $\bar{x}$ of x exists; let $\Delta > 0$ and $\tau \in \mathrm{R}$ be given. x is said to satisfy a *mean resonance condition with respect to* Δ *and* τ if and only if there exist rational numbers p_1, p_2, and p_3, with $p_1 \neq 0$, such that

$$p_1 \frac{\bar{x}}{\Delta} + p_2 \frac{\tau}{T} + p_3 = 0 \; . \tag{MRC}$$ □

Observe that "most" T-periodic x's *do not* satisfy a mean resonance condition with respect to Δ and τ; in fact, if both $\Delta^{-1}\bar{x}$ and τ/T are irrational, and their ratio is irrational, then (MRC) does not hold for any triad of $\{p_i\}$ of rational numbers.

Theorem 2.2, which is the main result of this section, states essentially that if $t \to x(t)$ is T-periodic and bandlimited, then $\tilde{G}$ is uniquely ergodic if and only if the input $t \to x(t)$ does not satisfy a mean resonance condition with respect to Δ and τ, where τ is the sampling interval and Δ is the bin width of the quantizer q. Theorem 3.2 is a partial generalization of Theorem 2.2 to inputs that are not bandlimited. We shall see that one can view both results as *averaging theorems;* they state that if the mean of the input leads to uniquely ergodic behavior, then so does the input itself.

2.2 Theorem: *Let* $t \to x(t)$, $t \in \mathrm{R}$, *be a bandlimited real-valued* T*-periodic signal; let* $\tau \in \mathrm{R}$ *be such that* τ *is not a rational multiple of* T, *and set* $\omega = 2\pi T^{-1}$. *The mapping* $\tilde{G}$ *from the torus to itself defined by*

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \xrightarrow{\tilde{G}} \begin{bmatrix} e^{i2\pi(\frac{1}{2}+\Delta^{-1}x(t))} z_1 \\ e^{i\omega\tau} z_2 \end{bmatrix} ,$$

with $z_2 = e^{i\omega t}$, $t \in [0,T)$, *is uniquely ergodic if and only if* x *does not satisfy a mean resonance condition with respect to* Δ *and* τ.

Proof: By T-periodicity and bandlimitedness, we can find an $N \geq 0$ and numbers $\{c_k\}$ for which

$$x(t) = \bar{x} + \sum_{k \neq 0, |k| \leq N} c_k e^{i\omega t} , \quad t \in [0,T] ,$$

where $\bar{x}$ is the mean of $t \to x(t)$. Denote by $x_o(t)$ the summation on the right-hand side of the last equation; define $r : [0,T] \to \mathrm{R}$ by

$$r(t) = \sum_{k \neq 0, |k| \leq N} \frac{2c_k}{\Delta(e^{ik\omega\tau} - 1)} e^{i\omega t} , \quad t \in [0,T] .$$

It is easy to check that $r(\langle t+\tau \rangle_T) - r(t) = 2\Delta^{-1} x_o(t)$ for every $t \in [0,T]$; hence if we set

$R(e^{i\omega t}) = (e^{i2\pi r(t)})$, then R satisfies Furstenberg's condition (F) with $m = 2$ and x_o plugged in for x.

A consequence of this calculation is that $t \to x_o(t)$ is a "bad" input to the sigma-delta modulator in the sense that the resulting mapping $\tilde{G}$ fails to be uniquely ergodic. It is not hard to check that the set of all inputs that are "bad" in the sense just described is closed under integer linear combinations; in particular, the difference between any two "bad" inputs is another "bad" input. It follows that $t \to x(t)$ is a "bad" input if and only if the constant input $\bar{x}$ is "bad." This, however, is true precisely when $\bar{x}$ satisfies a mean resonance condition with respect to Δ and τ. If $\bar{x}$ satisfies no such condition, the mapping $\tilde{G}$ arising from the constant input $\bar{x}$ is the product of two irrational rotations of the circle, where the "rotation angles" are not rationally related, and is therefore uniquely ergodic (see, for example, [7], Chapter II, Proposition 2.7 and Theorem 3.2). On the other hand, if $\bar{x}$ satisfies a mean resonance condition with respect to Δ and τ, then with the aid of (MRC) we can find integers n_1, n_2, and n_3 such that

$$n_1(½+\frac{\bar{x}}{\Delta})+n_2\frac{\tau}{T} = n_3$$

Define $r : [0,T] \to \mathrm{R}$ by $r(t) = n_2 t/T$ and set $R(e^{i\omega t}) = e^{i2\pi r(t)}$; then R satisfies (F) with $m = n_1$ and $\bar{x}$ substituted for $x(t)$, from which it follows that the constant input $\bar{x}$ is "bad." (In checking the last assertions, it's helpful to use the identity $R(e^{i\omega(t+\tau)}) = e^{i2\pi r(<t+\tau>_T)}$.) □

From Theorem 2.2 there follow some strong statements about the asymptotic behavior of the normalized error sequence $\{k \to e_k = \Delta^{-1}\varepsilon_k\}$ in the sigma-delta modulator of Figure 1. Suppose that $t \to x(t)$, $t \in \mathrm{R}$, is a bandlimited real-valued T-periodic signal satisfying $|x(t)| \le ½(M-1)\Delta$, and x_k in Figure 1 is $x(t_0+k\tau)$, where $t_0 \in \mathrm{R}$ is arbitrary and τ is not a rational multiple of T. Theorem 2.2 implies that so long as $\bar{x}$, the average value of $t \to x(t)$, does not satisfy a mean resonance condition with respect to Δ and τ, it is true that for arbitrary u_0 with $|u_0| < ½(M-1)\Delta$, $\{e(k)\}$ is asymptotically uniform in the sense that for every interval V contained in $[-\Delta, \Delta]$ and every $k \ge 0$, we have

$$\lim_{K \to \infty} \frac{1}{K} \sum_{l=0}^{K-1} 1_V(e(k)) = \lambda(V) ,$$

where λ is Lebesgue measure and 1_V is the indicator function of the interval V. Moreover,

$$\bar{e} \overset{\Delta}{=} \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} e_l = 0 , \tag{GGF1}$$

and, for every $k \ge 0$,

$$R_e(k) \overset{\Delta}{=} \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} e(l)e(l+k) = \frac{1}{12} - \frac{1}{2T}\int_0^T <\beta_k(t)>_1(1-<\beta_k(t)>_1)dt , \tag{GGF2}$$

where $<\ >_1$ denotes fractional part (mod1) and

$$\beta_k(t) = \frac{k}{2}+\frac{1}{\Delta}\sum_{l=0}^{k-1} x(t+l\tau) .$$

Since $e_k = ½-y_k$, (GGF1) is a direct consequence of the unique ergodicity of the mapping $\tilde{G}$ of Theorem 2.2; to see this, use $f(y) = y$ in (***). Formula (GGF2) comes from (***) with $f(y) = y[G^k(y)]_1$, $y \in [0,1]$, where G denotes the transformation of $[0,1] \times [0,T]$ that takes (y_k, t_k) to (y_{k+1}, t_{k+1}) and $[G^k(y)]_1$ is the first component of $G^k(y)$.

Our analysis so far has demonstrated that the *Generalized Gray's Formulas* (GGF1) and (GGF2) hold for "generic" bandlimited T-periodic inputs $t \to x(t)$, where "generic" translates as "having an average value that satisfies no mean resonance condition with respect to the quantizer bin width Δ and the sampling interval τ." Before moving on to §3, where we develop a version of Theorem 2.2 that holds for inputs that aren't bandlimited, it is worth commenting on one subtlety attending the relationship between unique ergodicity of the mapping $\tilde{G}$ and the existence of asymptotic statistics such as those in the Generalized Gray's Formulas. We are interested in the existence of asymptotic statistics for the y- and e-sequences; unique ergodicity of $\tilde{G}$ is only a *sufficient* condition for those asymptotic statistics to be well-defined and independent of the initializations y_0 and t_0. Suppose, for example, that $x(t) \equiv \bar{x}$, where $\Delta^{-1}\bar{x}$ is irrational but rationally related to τ/T. In this case, the mapping $\tilde{G}$ of Theorem 2.2 is *not* uniquely ergodic, but the asymptotic y-statistics are still well-defined and independent of initializations (c.f. our discussion of the constant-input case). This anomaly should not be alarming; the proof of Furstenberg's Theorem hinges on the fact that the existence of a measurable R satisfying condition (F) implies that the function $(z_1, z_2) \rightarrow R(z_2)z_1^{-m}$ is constant along trajectories of $\tilde{G}$. Suppose, for example, that $x(t) \equiv \bar{x}$ and $\Delta^{-1}\bar{x} = \tau/T-½$. It is easy to check that $R(z_2) = z_2$ satisfies (F) with $m = 1$; thus, the function $(z_1, z_2) \rightarrow z_2 z_1^{-1}$ is constant along trajectories of $\tilde{G}$. In terms of the y-sequence, all this means is that $y_k - t_k \equiv y_0 - t_0$, which doesn't affect the existence of (y_0, t_0)-independence of the y-statistics. The reader can check, however, that for non-constant bandlimited inputs whose average values satisfy a mean resonance condition, the invariant functions one obtains from Furstenberg's result are nontrivial, and would seem to preclude the existence of certain asymptotic y-statistics. Consequently, we regard the unique ergodicity of $\tilde{G}$ as "almost necessary" for the statistical well-behavedness of the y- and e-sequences.

3. A Preliminary Result for Broadband Inputs

Applying Furstenberg's results to inputs that are not bandlimited turns out to be a surprisingly delicate exercise in real analysis. It is our firm opinion that Theorem 3.2 below is *not* the strongest result that actually holds; we hope soon to have a proof of the following more general assertion.

3.1 Conjecture *With notation the same as in Theorem 2.2, suppose that $t \rightarrow x(t)$ is a T-periodic and piecewise continuous function whose jumps (if any) are integer multiples of Δ. Let $x_c : [0, T] \rightarrow \mathbf{R}$ be the unique continuous function that satisfies $x_c(0) = x(0)$ and $e^{i2\pi(x_c(t)-x(t))} = 1$ for every $t \in [0, T]$. The mapping $\tilde{G}$ is uniquely ergodic if the T-periodic extension of x_c does not satisfy a mean resonance condition with respect to Δ and τ.* ☐

Before describing what we actually *can* prove, it is instructive to consider what Furstenberg's results say about inputs that aren't bandlimited. In essence, we can say with some confidence that only rarely does a broadband T-periodic input satisfying the conditions in Conjecture 3.1 lead to a $\tilde{G}$ that is *not* uniquely ergodic. To see this, suppose $t \rightarrow x(t)$ is a zero-mean continuous T-periodic input that is not bandlimited and satisfies Furstenberg's condition (F) with some measurable R and $m = 2$. As in the proof of Theorem 2.2, we can find a bounded function $r : [0, T] \rightarrow \mathbf{R}$ so that $r(<t+\tau>_T) - r(t) = 2\Delta^{-1}x(t)$ for every $t \in [0, T]$. Let c_k be the kth Fourier coefficient of x and d_k the kth Fourier coefficient of r; we conclude that $d_k = 2\Delta^{-1}(1-e^{ik\omega\tau})^{-1}c_k$. The term multiplying c_k in the last identity gets arbitrarily large as $k \rightarrow \infty$, and does so somewhat erratically; thus, in order for $\{d_k\}$ to be bounded in k (which it must be), the $\{c_k\}$ sequence will have to be rather special.

Furstenberg shows in [3] that if condition (F) is satisfied by some R that is *not* equal almost everywhere to a continuous function, then some of the "ergodic averages" given by the left-hand side of (***) do not exist for every y_0 and t_0. This circumstance prompts us to examine first the question of when the input x is such that condition (F) can be solved for some $m \neq 0$ and a *continuous* R, or at least an R that can be modified on a set of measure zero so as to produce a continuous function R_c. The point is that if x is such that (F) can be solved for R and m, and R *cannot* be so modified, we know that some of the ergodic averages along trajectories of $\tilde{G}$ will not exist.

3.2 Theorem: *With assumptions and notation the same as Conjecture 3.1, suppose that $z_2 \to R(z_2)$ and $m \neq 0$ satisfy condition (F) along with $t \to x(t)$, and that R is equal almost everywhere to a continuous function R_c. Then the periodic extension of x_c satisfies a mean resonance condition with respect to Δ and τ.*

Proof: The hypotheses guarantee that (F) holds *almost everywhere* when R is replaced by R_c and x is replaced by x_c. Imitating the proof of Theorem 2.2, we can find a *continuous* function $r_c : [0,T] \to \mathrm{R}$ that satisfies $R_c(e^{i2\pi t}) = e^{i2\pi r_c(t)}$ for all $t \in [0,T]$ and $<r_c(<t+\tau>_T) - r_c(t) - m(\tfrac{1}{2}+\Delta^{-1}x_c(t))>_1 = 0$ for *almost* all $t \in [0,T]$. Since R_c is continuous, we know that $r_c(T) - r_c(0)$ is equal to some integer d. Observe next that the function

$$t \to r_c(<t+\tau>_T) - r_c(t) + d\,1_{[T-\tau,T]}(t) - m(\tfrac{1}{2}+\Delta^{-1}x_c(t))$$

is continuous on $[0,T]$ and is equal almost everywhere to an integer, from which it follows that it is equal *everywhere* to the same integer M. Integrating the equality

$$r_c(<t+\tau>_T) - r_c(t) + d\,1_{[T-\tau,T)}(t) - m(\tfrac{1}{2}+\Delta^{-1}x_c(t)) = M$$

over $[0,T]$ yields the mean resonance condition

$$-m\frac{\bar{x}_c}{\Delta} + d\frac{\tau}{T} = M + \frac{m}{2} ,$$

and the proof is complete. □

Since we have run out of space, it is impossible to elaborate further on Theorem 3.2 and its consequences. Suffice it to say that in addition to addressing Conjecture 3.1, we are currently working on extending our results to cover certain situations where the input $t \to x(t)$ to the sigma-delta modulator is *stochastic,* as well as to "non-ideal" situations where the discrete-time integrator appearing in Figure 1 is "leaky." We are also focusing some effort on the important problem of determining the *spectral* consequences of the Generalized Gray's Formulas. Furthermore, it turns out that Furstenberg's results apply nicely to *multi-stage* sigma-delta modulators (see [5]), and we expect soon to have generalizations of Theorems 2.2 and 3.2 to situations involving those and other oversampled ADCs that one can model as interconnections and modifications of systems such as the sigma-delta modulator in Figure 1.

Acknowledgements

This research was supported in part by the U. S. Army Research Office through the Mathematical Sciences Institute at Cornell University, Ithaca, NY, and was conducted while the author was visiting the Information Systems Laboratory at Stanford University. The author would like to thank Robert M. Gray for his warm hospitality during the author's visit.

REFERENCES

[1] Bennett, W. R., "Spectra of Quantized Signals." *Bell System Technical Journal,* Volume 27 (1948), pp. 446-472.

[2] Delchamps, David F. "Stabilizing a Linear System With Quantized State Feedback," to appear in *IEEE Transactions on Automatic Control,* Volume AC-35 (1990).

[3] Furstenberg, Harry, "Strict Ergodicity and Transformations of the Torus," *American Journal of Mathematics,* Volume 83 (1961). pp. 573-601.

[4] Gray, Robert M., "Oversampled Sigma-Delta Modulation," *IEEE Transactions on Communications,* Volume COM-35 (1987), pp. 481-489.

[5] Gray, Robert M., "Quantization Noise Spectra," to appear in *IEEE Transactions on Information Theory,* 1990.

[6] Kieffer, John, "Analysis of dc Input response for a Class of One-Bit Feedback Encoders," *IEEE Transactions on Communications,* Volume 38 (1990), pp. 337-340.

[7] Mañé, Ricardo. *Ergodic Theory and Differentiable Dynamics.* New York: Springer-Verlag, Inc., 1987.

OPTIMAL QUANTIZERS: COMPUTATIONAL IMPROVEMENT AND PERFORMANCE

STEVEN CRAIG GREER and JAMES GEORGE DUNHAM
Department of Electrical Engineering
Southern Methodist University
Dallas, Texas 75275

SUMMARY

Algorithms which decrease the computation time of step-by-step optimal quantizers by up to two orders of magnitude have been developed. The performance of the optimal quantizer is compared to two DPCM systems: one with a uniform quantizer and one with a quantizer designed to match the distribution of the input. Two sources are considered: 1'st order Gauss-Markov and 1'st order (truncated) Cauchy-Markov.

The theory of optimal step-by-step quantizers has previously been developed [1], [2] and applied to various sources. Optimal quantizers are based on a probability density function (p.d.f.) that is iteratively updated with every input sample. If x_n is the input sample at time n and y_n is the quantized sample at time n, the optimal quantizer is designed using the p.d.f. $p(x_n \mid y_0^{n-1})$ where y_0^{n-1} represents all quantized outputs up to time $n-1$. This p.d.f. is calculated using

$$p(x_n \mid y_0^{n-1}) = \int_{-\infty}^{\infty} p(x_{n-1} \mid y_0^{n-1})\, p(x_n \mid x_{n-1})\, dx_{n-1}. \qquad (1)$$

The quantizer decision and representation levels are calculated from $p(x_n \mid y_0^{n-1})$ using the Lloyd-Max quantization algorithm[3]. The p.d.f. $p(x_n \mid y_0^n)$ is then calculated for the next iteration using

$$p(x_n \mid y_0^n) = \frac{Q_{y_n}(x_n) \cdot p(x_n \mid y_0^{n-1})}{\int_{Q_{y_n}} p(x_n \mid y_0^{n-1})\, dx_n} \qquad (2)$$

where Q_{y_n} is an indicator function.

The implementation of the optimal quantizer is computationally intensive due to the integral equations and the iterative nature of the Lloyd-Max algorithm. The number of multiplies needed to numerically solve each integral equation is proportional to the square of the number of bins representing the p.d.f. The number of multiplies used to calculate the Lloyd-Max quantizer is proportional to the product of the number of bins representing the p.d.f. and the number of iterations of the Lloyd-Max algorithm.

The first improvement in computation time can be made by noting that equation (1) is of the form $p(x) = \int q(y) r(x - \alpha y) dx$. If we let $p'(x) = p(\alpha x)$ and $r'(x) = r(\alpha x)$ then $p'(x) = \int q(y) r'(x-y) dy$, a convolution. The Fast Hartley Transform (FHT) can be used to compute the transform in the frequency domain. As with the FFT approach, the computation time varies as $N \cdot log(N)$ rather than N^2. An interpolation between the points of $p'(x)$ must then be calculated to regain the density $p(x)$.

The second improvement in computation time is made to the Lloyd-Max algorithm. The usual implementation of the Lloyd-Max algorithm calculates, for each iteration,

$$centroid(k) = \frac{\sum_{i=l1}^{l2} x_i \cdot p(x_i)}{\sum_{i=l1}^{l2} p(x_i)} \text{ and } boundary(k) = \frac{centroid(k) + centroid(k-1)}{2} \quad k = 1, L$$

where L is the number of quantization levels and $l1$ and $l2$ are the first and last bins within the quantization region denoted by k. The total computation for the algorithm is then $M \cdot (N + L)$ multiplies, $M \cdot (2 \cdot N + L)$ additions and $L \cdot M$ divisions where M is the number of iterations of the algorithm and N is the number of discrete points representing the p.d.f.

This computation can be reduced significantly by precalculating two arrays and using their contents in the iterative updates of the centroids and boundaries of the quantization levels. In the first array, each array element j contains $A(j) = \sum_{i=0}^{j} x_i \cdot p(x_i)$ $j = 0, N-1$. In the second array, each array element j contains

$B(j) = \sum_{i=0}^{j} p(x_i)$. To compute a centroid given the boundaries of the previous iteration, compute $centroid(k) = (A(l2) - A(l1{-}1))/(B(l2) - B(l1{-}1))$ by accessing the appropriate elements of the two arrays containing these precomputed values. The boundaries are computed as before. The computation is then reduced to a precomputation of N multiplies and $2{\cdot}N$ additions and, for each iteration, $3{\cdot}L$ additions and $2{\cdot}L$ divisions. The processing for each iteration is then reduced by N multiplies and $2{\cdot}(N-L)$ additions. The number of iterations increases with the number of quantization levels and can go over 100 for 16 quantization levels.

The first table below gives the results of coding 10,000 samples of 1'st order Gauss-Markov data with Uniform DPCM (Unf), with Gaussian DPCM (Gau) and with the step-by step optimization (Opt) method. The second table below gives the results of coding 10,000 samples of 1'st order truncated Cauchy-Markov data with Uniform DPCM (Unf), with Cauchy DPCM (Chy) and with the step-by step optimization (Opt) method. The Cauchy density was truncated to plus and minus 5.74 standard deviations, retaining 96.8% of the density. Results indicate that the performance of the Gaussian DPCM quantizer is very close to that of the optimal quantizer, while improvement over DPCM can be obtained in the Cauchy-Markov case for low bit rates and high correlation.

GAUSS-MARKOV												
	1 BIT			2 BITS			3 BITS			4 BITS		
Alpha	Unf	Gau	Opt	Unf	Gau	Opt	Unf	Gau	Opt	Unf	Gau	Opt
.0	.365	.365	.365	.122	.120	.120	.0393	.0362	.0362	.0123	.0099	.0099
.1	.363	.363	.363	.121	.119	.119	.0388	.0357	.0357	.0122	.0099	.0099
.2	.356	.356	.356	.117	.116	.116	.0376	.0346	.0345	.0118	.0096	.0096
.3	.344	.344	.344	.111	.110	.110	.0357	.0328	.0328	.0112	.0091	.0091
.4	.327	.327	.327	.103	.102	.102	.0330	.0304	.0304	.0103	.0084	.0084
.5	.301	.301	.302	.094	.092	.092	.0295	.0274	.0273	.0092	.0075	.0075
.6	.268	.268	.269	.081	.080	.080	.0253	.0234	.0235	.0079	.0064	.0065
.7	.226	.226	.226	.066	.065	.066	.0201	.0187	.0188	.0063	.0051	.0051
.8	.171	.171	.172	.047	.046	.046	.0144	.0132	.0133	.0044	.0035	.0036
.9	.098	.098	.100	.025	.025	.025	.0077	.0070	.0070	.0024	.0019	.0019

CAUCHY-MARKOV												
	1 BIT			2 BITS			3 BITS			4 BITS		
Alpha	Unf	Chy	Opt	Unf	Chy	Opt	Unf	Chy	Opt	Unf	Chy	Opt
.0	.699	.699	.699	.337	.203	.203	.1186	.0569	.0569	.0344	.0147	.0147
.1	.692	.692	.692	.332	.200	.201	.1171	.0564	.0565	.0340	.0145	.0146
.2	.674	.674	.636	.320	.194	.194	.1132	.0547	.0547	.0328	.0141	.0141
.3	.646	.646	.595	.302	.186	.185	.1066	.0518	.0522	.0310	.0134	.0134
.4	.611	.610	.556	.278	.175	.172	.0970	.0481	.0484	.0285	.0123	.0124
.5	.563	.563	.514	.247	.160	.156	.0860	.0435	.0434	.0252	.0110	.0111
.6	.502	.503	.442	.208	.141	.136	.0729	.0372	.0374	.0213	.0094	.0095
.7	.418	.417	.367	.165	.117	.109	.0569	.0300	.0300	.0170	.0075	.0077
.8	.313	.317	.287	.115	.086	.077	.0396	.0215	.0215	.0120	.0053	.0054
.9	.180	.181	.159	.061	.049	.042	.0211	.0115	.0115	.0076	.0028	.0029

The table below shows timing results in seconds per sample using a DEC 3100 workstation.

Bits	Convolution		FHT		Speedup
	Time	# bins	Time	FHT size	Ratio
1	.97	400	.06	2048	16.2
2	3.7	800	.13	4096	28.5
3	14.5	1600	.29	8192	50.1
4	58.3	3200	.69	16384	84.5

[1] Dunham, J.G., "Optimal Discrete Time Delta Modulation System," I.E.E.E. Trans. Comm., vol. COM-34, No5, 510,512, May 1986

[2] Dunham, J.G. and Ghosh, A., "Improving DPCM System Performance," Proc. I.E.E.E. Global Telecom. Conf., pp. 1890-1894, Dallas, Tx. Nov. 1989

[3] Lloyd, S. P., "Least Squares Quantization in PCM," I.E.E.E. Trans. Inform. Theory, vol. IT-28, pp.129-137, Mar. 1982

Regularity of the Zero Dynamics Algorithm and Approximate Input Output Decoupling

M. Jankovic and B.K. Ghosh

Washington University
Department of Systems Science and Mathematics
Campus Box 1040 St. Louis, MO 63130

Abstract. An approximate decoupling algorithm is introduced in this paper that formalizes the procedure to obtain input-output decouplable approximate systems. It is also shown that the procedure preserves regularity of the zero dynamics and this fact has been used to prove an approximate tracking theorem under the assumption that the zero dynamics is asymptotically stable. This generalizes an earlier result due to Hauser, Sastry and Kokotovic wherein exponential stability of the zero dynamics is required.

Keywords. Nonlinear control, input output decoupling, dynamic extension, approximate linearization, zero dynamics

1. Introduction

In nonlinear control theory, an interesting problem is to obtain an approximate model which is in some sense close to a given system and satisfies certain desirable properties. For example, we may try to preserve as many higher order terms as possible, and to obtain a system which is internally linearizable or input output linearizable. One attempt in this direction has been made in [K] wherein a necessary and sufficient condition for internal linearization up to an order $p \geq 2$ is given. Another approach taken in [HSK] is to identify those terms that prevent the system to be input-output linearizable and to take them out, leaving the approximate system close to the original one. Simulations carried out on a simple "ball and beam" system clearly show the advantages of this method over the well known Jacobian linearization approach. In [GDBHS] this idea has been extended to MIMO systems as well.

This paper examines the behavior of the zero dynamics under approximation. Cases have been analyzed in which the regularity of the zero dynamics and the zero dynamics itself are preserved. Furthermore it is shown that under certain regularity conditions (namely that of ther zero dynamics algorithm, see [I] for definitions) and for a certain choice of approximating functions, the approximate tracking theorem ([HSK] Theorem 3.4) can be restated assuming only asymptotic stability instead of exponential stability of the zero dynamics.

The rest of this section introduces notations and contains some preliminary results. In section 2 an algorithm, based on dynamic extension, is introduced that gives a specific choice of approximating functions ϕ_i, ψ_i and also a state space description of the approximated system. The choice of functions ϕ_i and ψ_i is such that the only terms neglected are those that prevent the system from being regular in the sense introduced in [DBG]. In section 3 the behavior of the zero dynamics under approximation is examined and in section 4 the approximate tracking theorem is restated and proved.

Throughout this paper the formalism introduced in [DBG] and [DBGM] is used. The affine nonlinear system is given by:

$$\Sigma \quad \begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{1.1}$$

where $x \in X \subset R^n$, $u \in U \subset R^m$, $y \in Y \subset R^m$. For simplicity the system is assumed to be square, i.e. have the same number of inputs and outputs. All functions and vector fields are assumed to be analytic functions of its arguments. Also

$$y^{(k)}(x,u,\dots,u^{(k-1)}) = \frac{\partial y^{(k-1)}}{\partial x}(f(x)+g(x)u) + \sum_i \frac{\partial y^{(k-1)}}{\partial u^{(i)}} u^{(i+1)}$$

may be viewed as a function mapping $X \times T^{k-1}U$ into R^m. In the same fashion one can define a sequence of canonical maps F_k and E_k : $F_0 : X \to Y$ by $y = h(x)$ and $E_0 : X \to X$ the identity map;

$$\text{for } k \geq 1 F_k : X \times T^{k-1}U \to T^kY \text{ by } (x,u,\dots,u^{(k-1)}) \to (y,\dots,y^{(k)}) \text{ and}$$
$$E_k : X \times T^{k-1}U \to X \times T^kY \text{ by } (x,u,\dots,u^{(k-1)}) \to (x,y,\dots,y^{(k)}).$$

T^kU is the kth order tangent bundle of U. Let K denote the field of rational functions of the components $(u,\dots,u^{(n-1)})$ with coefficients that are meromorphic in x (see [DBGM] for details).

Consider the set of indices defined by $\sigma_i = \text{rank}_K E_i - \text{rank}_K E_{i-1}$ $i = 1,\dots,n$. Each σ_i corresponds to the number of zeros at infinity of order less than or equal to i. The structure at infinity, denoted by $S_\infty(\Sigma)$, of the nonlinear system Σ is a set $\{\sigma_1,\dots,\sigma_n\}$. The structure at infinity plays a crucial role in the input-output decoupling of nonlinear systems. Denote by $S_\infty(\Sigma^L)$ the structure at infinity of the Jacobian linearization around x_0 of Σ. In [GDBHS] it was shown that the local regularity of the equilibrium, in the sense of [DBG], is equivalent to having $S_\infty(\Sigma^L) = S_\infty(\Sigma)$ in a neighborhood of x_0. Next result shows that right invertibility of linear approximation is sufficient for the existence of functions $\phi_k(x,u,\dots,u^{(k-1)}$ and $\psi_k(x,u,\dots,u^{(k-1)})$ $0 \leq k \leq n$ (Proposition 3.2 [GDBHS]).

Proposition 1.1: Suppose x_0 is an equilibrium point and that Σ_L is right invertible. Then there exist smooth functions $\phi_0(x)$, $\psi_0(x)$ and $\phi_k(x,u,..,u^{(k-1)})$, $\psi_k(x,u,\dots,u^{(k-1)})$ for $1 \leq k \leq n$ where ψ_k is of the second or higher order in its arguments, such that

$$\begin{aligned} (1.1)\qquad & h(x) = \phi_0(x) + \psi_0(x) \\ & \phi_i^{(1)}(x,u,\dots,u^{(i-1)}) = \phi_{i+1}(x,\dots,u^{(i)}) + \psi_{i+1}(x,\dots,u^{(i)}) \end{aligned}$$

$$(1.2)\qquad \dim \text{span}_R\{d\phi_i(x_0,0,\dots,0), 0 \leq i \leq k\} = m(k+1)$$

$$(1.3)\qquad \dim \text{span}_K\{dx, d\phi_i(x,\dots,u^{(i-1)}), 1 \leq i \leq k\} = n + \sigma_1^L + \dots + \sigma_k^L$$

Definition 1.2: A set of functions $\{\phi_k, \psi_k, 0 \leq k \leq n\}$ is said to satisfy condition A1 if it satisfies conditions (1.1), (1.2), (1.3) and ψ_k's are of order 2 or higher.

Proposition 1.1 allows us to construct a normal form for the approximate system using dynamic extension ([GDBHS] Theorem 3.4). From now on it will be assumed that the equilibrium point is 0.

2. Approximate Decoupling Algorithm (ADA)

In general there are many different ways to choose the functions ϕ_k and ψ_k and in here a brief description of the one that utilizes dynamic extension philosophy (see

[DBG]) is given. This algorithm formalizes the procedure used in warehouse crane example ([GDBHS] section 4) to obtain an approximate model.

Suppose Σ^L is right invertible and its structure at infinity is given by $\{\sigma_1^L, \dots, \sigma_n^L\}$. Choose $\phi_0(x) = h(x) = y$ and $\psi_0(x) = 0$ and note that they satisfy conditions (A1).

Step 1: Calculate $\phi_0^{(1)} = a(x) + b(x)u$ and note that the rank$_R b_1(0) = \sigma_1^L$. By renumerating the outputs if necessary we partition

$$\phi_0^{(1)}(x,u) = \begin{bmatrix} \bar{a}_1(x) + \bar{b}_1(x)u \\ \hat{a}_1(x) + \hat{b}_1(x)u \end{bmatrix}$$

where rank$_R \bar{b}_1(0)$ =rank$_R b_1(0) = \sigma_1^L$.

There exist two functions $\alpha_1(x)$ and $\beta_1(x)$, where $\beta_1(x)$ is $m \times m$ matrix invertible over R, such that by applying the feedback $u = \alpha_1(x) + \beta_1(x)v_1$ and partitioning $v_1 = \mathrm{col}(\bar{v}_1, \hat{v}_1)$ we get

$$\phi_0^{(1)}(x,u) = \begin{bmatrix} \bar{v}_1 \\ \hat{c}_1(z) + \hat{d}_1(x)\bar{v}_1 + \delta_1(x)\hat{v}_1 \end{bmatrix} \quad (\dim \bar{v}_1 = \sigma_1^L)$$

Knowing that

$$\frac{\partial \phi_0^{(1)}(x,v)}{\partial v_1} = \begin{bmatrix} I_{\sigma_1^L} & 0 \\ \hat{d}_1(x) & \delta_1(x) \end{bmatrix}$$

where $I_{\sigma_1^L}$ is $\sigma_1^L \times \sigma_1^L$ identity matrix) and that

$$\mathrm{rank}_R \frac{\partial \phi_0^{(1)}(0,0)}{\partial v_1} = \sigma_1^L$$

we conclude that $\delta_1(0) = 0$ or in other words $\delta_1(x) = O(x)$, and since that term destroys regularity it has to go out.

Therefore choose $\psi_1(x,v) = \delta_1(x)\hat{v}_1$ and $\phi_1(x,v) = \begin{bmatrix} \bar{v}_1 \\ \hat{c}_1(x) + \hat{d}_1(x)\bar{v}_1 \end{bmatrix}$ where col($\bar{v}_1 \hat{v}_1$) $= \beta_1^{-1}(x)(u - \alpha_1(x))$. Since by invertibility assumption dim span$_R\{dy_L^{(1)}\} = m$ we conclude that dim span$_R\{d\phi_i(0,0) i = 0,1\} = 2m$ and dim span$_K\{dx, d\phi_1(x,u)\} = n + \sigma_1^L$. Therefore ϕ_1 and ψ_1 satisfy (A1). Now we add integrators on the input channels $\bar{v}_1$: $\begin{matrix} \dot{\bar{v}}_1 = \bar{u}_1 \\ \hat{v}_1 = \hat{u}_1 \end{matrix}$.

Step 2: differentiate $\phi_1(x, \bar{v}_1)$ to obtain

$$\phi_1^{(1)}(x, \bar{v}_1, u_1) = \begin{bmatrix} \bar{u}_1 \\ a_2(x, \bar{v}_1) + b_2(x, \bar{v}_1)u_1 \end{bmatrix}$$

and now rank$_R \dfrac{\partial \phi_1^{(1)}(0,0,0)}{\partial u_1} = \sigma_2^L$. Partition $b_2(x, \bar{v}_1)$, after renumerating the outputs if necessary, into col($\bar{b}_2(x, \bar{v}_1)$, $\hat{b}_2(x, \bar{v}_1)$) such that

$$\mathrm{rank}_R \begin{bmatrix} [I \quad 0] \\ \bar{b}_2(x, \bar{v}_1) \end{bmatrix} = \sigma_2^L$$

and I and 0 are identity and zero matrices of appropriate dimensions. Now there exists a feedback $u_1 = \alpha_2(x, \bar{v}_1) + \beta_2(x, \bar{v}_1)v_2$ where β_2 is invertible over R such that $\phi_1^{(1)}$, which is now function of $x, \bar{v}_1$ and v_2, takes the form:

$$\phi_1^{(1)}(x, \bar{v}_1, v_2) = \begin{bmatrix} \bar{v}_2 \\ \hat{c}_2(x, \bar{v}_1) + \hat{d}_2(x, \bar{v}_1)\bar{v}_2 + \delta_2(x, \bar{v}_1)\hat{v}_2 \end{bmatrix}$$

where v_2 is partitioned into $\mathrm{col}(\bar{v}_2, \hat{v}_2)$ and the dimension of $\bar{v}_2$ is σ_2^L. Since $\mathrm{rank}_R \dfrac{\partial \phi_1^{(1))}(0,0,0)}{\partial u_1} = \sigma_2^L$ we conclude that $\delta_2(0,0) = 0$ i.e. that δ_2 is $O(x, \bar{v}_1)$. The choice of functions ϕ_2 and ψ_2 is obvious:

$$\phi_2(x, \bar{v}_1, \bar{v}_2) = \begin{bmatrix} \bar{v}_2 \\ \hat{c}_2(x, \bar{v}_1) + \hat{d}_2(x, \bar{v}_1)\bar{v}_2 \end{bmatrix}$$

and $\psi_2(x, \bar{v}_1, \hat{v}_2) = \mathrm{col}(0, \delta_2(x, \bar{v}_1)\hat{v}_2$. We can also compute ϕ_2 and ψ_2 as functions of x, u, u knowing that:

$$\begin{aligned}
&\bar{v}_1 = [I:0]\beta^{-1}(x)(u - \alpha_1(x)) \text{ where } I \text{ is } \sigma_1^L \times \sigma_1^L \text{ identity matrix} \\
&\dot{\bar{v}}_1 = [I:0]\{\dot{\beta}_1^{-1}(x)(u - \alpha_1(x)) + \beta_1^{-1}(x)(\dot{u} - \dot{\alpha}_1(x))\} \\
&u_1 = \mathrm{col}(\dot{\bar{v}}_1, \hat{v}_1) \\
&\bar{v}_2 = [I\,0]\beta^{-1}(x)(u_1 - \alpha_2(x)) \text{ where } I \text{ is } \sigma_2^L x \sigma_2^L \text{ identity matrix} \\
&\hat{v}_2 = [0\,I]\beta_2^{-1}(x)(u_1 - \alpha_2(x)) \text{ where } I \text{ is } (m - \sigma_2^L) \times (m - \sigma_2^L) \text{ identity matrix}
\end{aligned}$$

One can easily check that ϕ_2 and ψ_2 together with $\phi_0, \phi_1, \psi_0, \psi_1$ satisfy condition (A1). We now add integrators on the set of input channels $\bar{v}_2$: $\begin{matrix} \dot{\bar{v}}_2 = \bar{u}_2 \\ \dot{\hat{v}}_2 = \hat{u}_2 \end{matrix}$ and proceed by differentiating ϕ_2.

We stop at step k^* when $\mathrm{rank}_R \dfrac{\partial \phi_{k^*-1}^{(1)}(0, \ldots, 0)}{\partial u_{k^*-1}} = \sigma_k^L * = m$ and the existence of such k^* is assured by right invertibility of the linear approximation. Then we choose ϕ_k* to be equal to $\phi_{k^*-1}^{(1)}$ and ψ_{k^*} to be zero.

Since $\dim \mathrm{span}_K\{d\phi_0, \ldots, d\phi_{k^*-1}\} = mk^*$ it is possible to select a set of independent coordinates $\xi_k = \phi_{k-1}$ $1 \leq k \leq k^*$ such that with the complementary set of coordinates η the system takes the form:

$$\begin{aligned}
\dot{\xi}_1 &= \xi_2 + \psi_1(\xi, \eta, v) \\
\dot{\xi}_2 &= \xi_3 + \psi_2(\xi, \eta, v) \\
&\vdots \\
\dot{\xi}_k* &= v + \psi_k * (\xi, \eta, v) \\
\dot{\eta} &= q(\xi, \eta, v) \\
y &= \xi_1 + \psi_0(\xi, \eta)
\end{aligned}$$

where in our case ψ_0 and ψ_k* are 0 but are, in general, different then 0 if some other approximating functions are chosen.

3. Approximation and regularity of the zero dynamics algorithm

Recall the zero dynamics algorithm (for reference see [I] section 6.1 and [DBG]). It is assumed that the zero dynamics is regular. Define $M_0 = \{x \in X | h(x) = 0\}$ and denote by M_0^c the connected component of M_0 which contains point 0. Shrink M_0^c to be defined only on a small neighborhood of 0 if necessary to have a constant dimension. Calculate

$$y^{(1)} = c_1(x) + d_1(x)u = \begin{bmatrix} \bar{c}_1(x) + \bar{d}_1(x)u \\ \hat{c}_1(x) + \hat{d}_1(x)u \end{bmatrix} = \begin{bmatrix} \bar{y}_1 \\ \hat{y}_1 \end{bmatrix}$$

where $y^{(1)}$ is partitioned such that $\mathrm{rank}_R d_1(0) = \mathrm{rank}_R \bar{d}_1(0) = r_1$. Regularity of the Zero Dynamics Algorithm implies that the $\mathrm{rank}_R d_1(x)$ is constant on M_0^c. Therefore there exist $R_1(x)$ such that

$$\hat{d}_1(x) = R_1(x)\bar{d}_1(x)$$

on M_0^c. Write $\hat{y}_1 = \hat{c}_1 + R_1(x)(\bar{y}_1 - \bar{c}_1(x)) \hat{=} \Theta_1(x, \bar{y}_1)$. Define $M_1 = \{x \in M_0^c | \Theta_1(x, \bar{y}_1) = 0\}$ and at each subsequent step repeat the procedure similar to the above. Note that regularity condition implies that at each step k the matrix

$$D_k(x, \bar{y}_1, \dots, \bar{y}_{k-1}) = \begin{bmatrix} \bar{d}_1(x) \\ \vdots \\ d_k(x, \bar{y}_1, \dots, \bar{y}_{k-1}) \end{bmatrix}$$

has a constant rank on M_{k-1}^c. For more details see [DBG].

In [I], page 389, it was shown that the zero dynamics is left unchanged under dynamics extension. Of course it is not the case with the regularity which is in general not preserved under dynamic extension as the following example shows:

Example 3.1:

$$\begin{aligned} \dot{x}_1 &= x_3 u_1 + u_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= u_1 \\ y_1 &= x_1 \\ y_2 &= x_2 \end{aligned}$$

This system has regular zero dynamics but addition of an integrator on the second input channel $\dot{u}_2 = v_2$ destroys regularity (becuase the matrix $D_1 = \begin{bmatrix} x_3 & 0 \\ 0 & 0 \end{bmatrix}$ does not have a constant rank on M_0^c.

The following two propositions show that the regularity of the zero dynamics is preserved under transformations (i.e. regular static state feedback and addition of integrators on input channels) used in ADA to obtain the canonical form (2.4).

Proposition 3.2: Suppose that Σ has the regular zero dynamics. Then the application of a static state feedback $u = \alpha(x) + \beta(x)v$ (with $\beta(x)$ invertible over R) preserves the regularity property of the zero dynamics and the two zero dynamics are the same. Proof of this proposition is trivial.

Denote by Σ_i $i = 0, \dots, k^*$ the sequence of systems obtained after ith step of ADA, where Σ_0 is by definition equal to Σ. Also denote by X_i and Z_i^* the corresponding state space and zero dynamics submanifolds for the system Σ_i. Then the following result holds:

Proposition 3.3: If the system Σ_i has regular zero dynamics so does Σ^*_{i+1} and Z^* is diffeomorphic to Z^*_{i+1}. Moreover if we extend Z^*_i to X_{i+1} then Z^*_{i+1} is a submanifold of Z^*_i on X_{i+1}.

Proof of 3.3: It suffices to consider only Σ_0 and Σ_1. In view of Proposition 3.2 without loss of generality assume that the feedback which decouples first δ^L_1 inputs and outputs has been applied. Denote by M^i_0 and M^i_1 the submanifolds obtained after ith step of zero dynamics algorithm corresponding to Σ_0 and Σ_1. So $M^0_0 = \{x \in X_0 | h(x)\}$ and $M^0_1 = \{x \in X_1 | h(x) = 0\}$.

Regularity implies that M^0_0 and M^0_1 have constant dimension in a neighborhood of 0 (say on $M^0_0 \cap \mathrm{U}^0_0$ and $M^0_1 \cap \mathrm{U}^0_1$). To simplify notation we will also denote by M^0_0 and M^0_1 the connected components of $M^0_0 \cap \mathrm{U}^0_0$ and $M^0_1 \cap \mathrm{U}^0_1$.

Now differentiate y with respect to the dynamics of Σ:

$$y^{(1)} = \begin{bmatrix} \bar{y}_1 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} \bar{u} \\ \bar{c}_1(x) + \hat{d}_1(x)\bar{u} + \delta_1(x)\hat{u} \end{bmatrix}$$

when, as before, $\dim \bar{u} = \sigma^L_1$. Regularity assumption (namely $\operatorname{rank}_R L_g h(x)$ is constant on M^0_0) implies that $\delta_1(x)$ being zero at the origin must also be zero on M^0_0.

Therefore $\Theta^1_0(x, \bar{y}_1) \hat{=} \hat{c}_1(x) + \hat{d}_1(z)\bar{y}_1$ and $M^1_0 = \{x \in X_0 | h(x) = 0, \Theta_1(x, 0) = 0\}$.

Now we repeat the same for Σ_1

$$y^{(1)} = \begin{bmatrix} \bar{y}_1 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} \bar{v}_1 \\ \hat{c}_1(x) + \hat{d}_1(x)\bar{v}_1 + d_1(x)\hat{u}_1 \end{bmatrix}$$

Note that $\bar{v}_1$ represents state variables in this case. Since $\delta_1(x) = 0$ on M^0_0 we conclude that it is 0 on M^0_1 and therefore the regularity is preserved after the first step. If follows that:

$$\Theta^1_1(x_1 \bar{v}_1) \hat{=} \begin{bmatrix} \bar{v}_1 \\ \hat{c}_1(x) + \hat{d}_1(x)\bar{v}_1 \end{bmatrix} \text{ and}$$

$$\begin{aligned} M^1_1 &= \{x \in X_1 | h(x) = 0 \ \bar{v}_1 = 0 \ \Theta_1(x, \bar{v}_1) = 0\} = \\ &= \{x \in X_1 | h(x) = 0 \ \bar{v}_1 = 0 \ \Theta_1(x, 0) = 0\}. \end{aligned}$$

Therefore M^1_1 and M^0_1 are diffeomorphic. Also if we extend M^1_0 on X_1 it follows that

$$M^1_1 = \{x \in M^1_0 | \bar{v}_1 = 0\}$$

and therefore it is a submanifold of M^1_0.

Now we go back to the original system Σ_0 and differentiate $\Theta^1_0(x, \bar{y}_1)$ to obtain:

$$\begin{aligned} \Theta^{1(1)}_0(x_1 \bar{y}_1) &= L_f \Theta^1_0 + \frac{\partial \Theta^1_0}{\partial \bar{y}_1} \bar{y}^{(1)}_1 + L_g \Theta^1_0 u = \\ &= a_2(x, \bar{y}_1, \bar{y}^{(1)}_1) + b_2(x_1 \bar{y}_1)u \end{aligned}$$

It follows

$$\operatorname{rank}_R \begin{bmatrix} \begin{bmatrix} I_{\sigma^L_1} & 0 \\ \hat{d}_1(0) & 0 \end{bmatrix} \\ b_2(0,0) \end{bmatrix} = \sigma^L_2.$$

Recall this matrix from ADA. Now partition $b_2(x, \bar{y}_1)$ into two parts: $b_2(x_1, \bar{y}_1) = \begin{bmatrix} \bar{b}_2(x_1, \bar{y}_1) \\ \hat{b}_2(x_1, \bar{y}_1) \end{bmatrix}$ such that $\operatorname{rank}_R \begin{bmatrix} [I_{\sigma_1^L} \quad 0] \\ b_2(0,0) \end{bmatrix} = \operatorname{rank}_R \begin{bmatrix} [I_{\sigma^L} \quad 0] \\ \bar{b}_2(0,0) \end{bmatrix} = \sigma_2^L$ and $\bar{b}_2(x_1, \bar{y}_1)$ has $\sigma_2^L - \sigma_1^L$ components.

There exists a matrix $R_0^2(x_1 \bar{y}_1)$ such that

$$R_0^2(x_1, \bar{y}_1) \begin{bmatrix} [I_{\sigma_1^L} \quad 0] \\ \bar{b}_2(x_1, \bar{y}_1) \end{bmatrix} = \hat{b}_2(x_1, \bar{y}_1)$$

and then $\Theta_0^2(x_1, \bar{y}_1, \bar{y}_1^{(1)}, \bar{y}_2) \mathrel{\hat{=}} \hat{a}_2(x_1, \bar{y}_1, \bar{y}_1^{(1)}) + R_0^2(x_1, \bar{y}_1) \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 - \bar{a}_2(x_1, \bar{y}_1, \bar{y}_1^{(1)}) \end{bmatrix}$ where partition $\Theta_0^{1(1)} = \begin{bmatrix} \bar{y}_2 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} \bar{a}_2 + \bar{b}_2 u \\ \hat{a}_2 + \hat{b}_2 u \end{bmatrix}$ corresponds to the one of b_2.

Now $M_0^2 = \{x \in X_0 | h(x) = 0, \Theta_0^1(x, 0) = 0,\ \Theta_0^2(x, 0, 0) = 0\}$.
We go to Σ_1 one more time to calculate

$$\Theta_1^{1(1)}(x_1 \bar{v}_1) = \begin{bmatrix} \bar{u}_1 \\ L_f \Theta_0^1 + \frac{\partial \Theta_0^1}{\partial \bar{v}_1} \bar{u}_1 + L_g \Theta_0^1 u_1 \end{bmatrix}$$

where $\Theta_0^1(x_1, \bar{v}_1) = \Theta_0^1(x, \bar{y}_1)|_{\bar{y}_1 = \bar{v}_1}$.

It is easy to see that $L_g \Theta_0^1(x, \bar{v}_1) u_1 = b_2(x, \bar{v}_1) u_1$. Again the regularity condition is satisfied because

$$\operatorname{rank}_R \frac{\partial}{\partial u_1} \begin{bmatrix} 0 \\ \bar{u}_1 \\ \frac{\partial \Theta_0^1}{\partial \bar{v}_1} \bar{u}_1 + b_2 u_1 \end{bmatrix} = \operatorname{rank}_R \begin{bmatrix} [I_{\sigma_1^1} \quad 0] \\ b_2(x, \bar{v}_1) \end{bmatrix} = \text{const.} = \sigma_2^L.$$

Partition $b_2(x, \bar{v}_1)$ as before and choose R_1^2 to be

$$R_1^2 = [-R_0^{22} \bar{\gamma}_1 + \hat{\gamma}_1 + R_0^{21} \vdots R_0^{22}]$$

where

$$\gamma_1 = \frac{\partial \Theta_0^1}{\partial \bar{v}_1}$$

is partitioned into $\begin{bmatrix} \bar{\gamma} \\ \hat{\gamma} \end{bmatrix}$ according to the partition of b_2 and where $R_0^2 = [R_2^{21} \vdots R_0^{22}]$ corresponds to the partition of $\begin{bmatrix} [I_{\sigma_1^L} \quad 0] \\ \bar{b}_2 \end{bmatrix}$.

Simple calculation shows that

$$R_1^2 \begin{bmatrix} [I_{\sigma_1^L} \quad 0] \\ \bar{\gamma}_1 + \bar{b}_2 \end{bmatrix} = \hat{\gamma}_1 + \hat{b}_2$$

Therefore $\Theta_1^2 \mathrel{\hat{=}} \hat{\gamma} + R_1^2 \begin{bmatrix} \bar{y}_{21} \\ \bar{y}_{22} - \bar{\kappa} \end{bmatrix}$ where $\kappa = L_g \Theta_0^1$ is partitioned into $\operatorname{col}(\bar{\kappa}, \hat{\kappa})$ according to the partition of b_2 and $\Theta_1^{1(1)} = \operatorname{col}(\bar{y}_{21}, \bar{y}_{22}, \hat{y}_2)$ where $\dim \bar{y}_{21} = \sigma_1^L \dim \bar{y}_{22} = \sigma_2^L - \sigma_1^2$.

Now if we put all pieces together we get

$$\Theta_1^2(x_1 \bar{v}_1, \bar{y}_2) = \hat{a}_2(x, \bar{v}_1, \bar{y}_{21}) + R_1^2(x, \bar{v}_1) \begin{bmatrix} \bar{y}_{21} \\ \bar{y}_{22} - \bar{a}_2(x, \bar{v}_1, \bar{y}_{21}) \end{bmatrix}$$

and $M_1^2 = \{x \in X_1 | h(x) = 0, \bar{v}_1 = 0, \Theta_1^1(x, \bar{v}_1) = 0, \Theta_1^2(x, \bar{v}_1, 0) = 0\} =$ $= \{x \in X_1 | h(x) = 0, \bar{v}_1 = 0, \Theta_0^1(x, 0) = 0, \Theta_0^2(x, 0, 0) = 0\}$. Now since we can relate $\bar{v}_1$ to $\bar{y}_1$, $\bar{y}_{21}$ to $\bar{y}_1^{(1)}$ and $\bar{y}_{22}$ to $\bar{y}_2$ we conclude that two algorithms are parallel from this point and the proposition is proved.

Recall from [GDBHS] theorem 3.4 that for any choice of functions $\phi_k(x, u, \ldots, u^{(k-1)})$, $\psi_k(x, u, \ldots, u^{(k-1)}) 0 \leq k \leq n$ that satisfy (A1) it is possible, by dynamic extension, to represent the system in the form (2.4) with different functions ψ_k and η. Certainly, ξ_k then corresponds to that choice of approximating functions ϕ_k.

Corollary 3.4: For any choice of functions ϕ_k, ψ_k $0 \leq k \leq n$ the corresponding normal form (2.4) has the regular zero dynamics if the original system has and the two zero dynamics are diffeomorphic.

Proof: For any choice of approximating functions ϕ_k, ψ_k $k = 1, \ldots, k^*$ the corresponding state space description in normal form has been obtained by application of the following transformation on the original system: static state feedback, addition of integrators on the input channels and change of variables (see that proof of Theorem 3.4 in [GDBHS]). Therefore it suffices to prove that for any choice of ϕ_k, ψ_k we may, at each step, select the same set of inputs to add integrators on. Denote by y_L the output of the linear approximation. Therefore $\dfrac{\partial y_L^{(k)}}{\partial u}$ are constant matrices. Now our result follows from the fact that $\dfrac{\partial \phi_k^{(1)}(0, \ldots, 0)}{\partial u} = \dfrac{\partial y_L^k}{\partial u}$ if ϕ_k is represented as a function of $x, u, \dot{u}, \ldots, u^{(k-1)}$. Therefore at each step we can select the same set of input channels that span σ_{k+1}^L independent rows to add integrators on regardless of the choice of ϕ_k.

Remark 3.5: Proposition 3.4 does not say that the regularity of the zero dynamics of Σ implies that of the approximate system. In general neglecting terms ψ_i may lead to having the zero dynamics of Σ and that of the approximate system with even different dimensions.

4. Approximate tracking

In this section we use results established above and some special properties of the normal form obtained from Approximate Decoupling Algorithm to restate Theorem 3.4 from [HSK] relaxing exponential stability assumption to only asymptotic stability. We start again with the normal form (2.4) obtained from Approximate Decoupling Algorithm. Denote by Z^* the zero dynamics manifold of the extended system (2.4).

Proposition 4.1 Suppose that the functions ϕ_k and ψ_k $0 \leq k \leq n$ and the normal form (2.4) are chosen as in Approximate Decoupling Algorithm and suppose that the original system has regular zero dynamics. Then the functions $\psi_i(\xi, \eta, v) = 0$ $1 \leq i \leq k^* - 1$ on Z^*.

Proof: From Proposition 3.3 we conclude that Σ_{i+1} has regular zero dynamics. At the $i + 1$th step of the zero dynamics algorithm when we differentiate Θ_i we get $\dfrac{\partial \Theta_i^{(1)}(x, \bar{v}_1, \ldots v_{i-1}, u_i)}{\partial u_i} = \begin{bmatrix} 0 & 0 \\ 0 & \delta_i(x, \ldots, \bar{v}_{i-1}) \end{bmatrix}$. The regularity now implies that $\delta_i(x, \ldots, \bar{v}_{i-1}) = 0$ on M_i and therefore on Z_i^*. In view of Proposition 3.3 $\delta_i(x, \ldots, \bar{v}_{i-1})$ $= 0$ on Z^*. Recall that at the ith step of ADA ψ_i has been chosen to be $\psi_i(x, \ldots, \bar{v}_{i-1}, v_i)$ $= \delta_i(x, \ldots, \bar{v}_{i-1})\hat{v}_i$. Therefore $\psi_i = 0$ on Z^*.

From Proposition 4.1 follows that $Z^* = \{(\xi, \eta) \in X | \xi_1 = \xi_2 = \ldots = \xi_{k^*} = 0\}$ The zero dynamics is said to be asymptotically stable if the subsystem $\dot{\eta} = q(0, \eta, 0)$ is asymptotically stable. Denote by v_i the ith component of the input vector v, and by c^i

the row vector $[c_k^i*, \dots, c_1^i]$ of the coefficients of a Hurwitz polynomial for $1 \le i \le m$. Denote by y_{di} the value of the ith component of the reference output. We may now use the above result to prove the following theorem.

Theorem 4.2: Suppose that the system is represented in the normal form (2.4) obtained by Approximate Decoupling Algorithm and that the original system (1.1) has regular zero dynamics which is asymptotically stable. Take any $\varepsilon > 0$. If the values of $y_d, y_d^{(1)}, \dots, y_d^{(k^*)}$ are sufficiently small, the input

$$v_i = y_{di}^{(k^*)} + c^i \begin{bmatrix} y_{di}^{(k^*-1)} - \xi_{k^*}^i \\ \vdots \\ y_{di} - \xi_1^i \end{bmatrix} \qquad 1 \le i \le m \tag{4.1}$$

guarantees that the states of the closed loop system remain in $B_\varepsilon(0)$. If the values of the reference output $y_d, y_d^{(1)}, \dots, y_d^{(k^*)}$ are in addition bounded in $B_\varepsilon(0)$, which in our case is almost not a constraint, then the tracking error is $O(\varepsilon^2)$.

Proof: The main problem is to show the boundedness of the states when the system (2.4) is driven by the input (4.1). We can partition q into $q(\xi, \eta, v) = q_1(\xi, \eta) + q_2(\xi, \eta)v$. Note then that q is affine and ψ_i's are linear functions in v. Partition the input in the obvious way $v = v(\bar{y}_d) + v(\xi)$ and suppose that only $v(\xi)$ has been applied. The system takes the form:

$$\begin{aligned} \dot{\xi} &= A\xi + \bar{\psi}(\xi, \eta, v(\xi)) \\ \dot{\eta} &= q(\xi, \eta, v(\xi)) \end{aligned} \tag{4.2}$$

By assumptions we know that $\dot{\eta} = q(0, \eta, 0)$ is asymptotically stable and A is Hurwitz. Also by Proposition 4.1 $\bar{\psi}(0, \eta, 0) = 0$, because it is 0 on Z^*, and since it contains only higher order terms we have $\dfrac{\partial \bar{\psi}(0,0)}{\partial \xi} = 0$. Therefore by Lemma from [I] page 442 we conclude that (4.2) is asymptotically stable. Now if we add $v(\bar{y}_d)$ the system may be written in the form:

$$\dot{z} = \bar{\gamma}(z) + p(z, \bar{y}_d) \quad \text{where } z = \begin{bmatrix} \xi \\ \eta \end{bmatrix}$$

$$\begin{aligned} &\bar{\gamma}(z) = \mathrm{col}(A\xi + \bar{\psi}(\xi, \eta, v(\xi)), q_1(\xi, \eta) + q_2(\xi, \eta)v(\xi)) \text{ and} \\ &p(z) = \mathrm{col}(\bar{\psi}(\xi, \eta, v(\bar{y}_d)), q_2(\xi, \eta)v(\bar{y}_d))) \end{aligned}$$

We already know that the system $\dot{z} = \bar{\gamma}(z)$ is asymptotically stable. Therefore from the total stability Lemma (see [H] or [I] page 444) the whole system is bounded if the driving term $p(z, \bar{y}_d)$ is sufficiently small and that is satisfied if the reference output $\bar{y}_d$ is small, since $p(z, \bar{y}_d)$ is linear in $\bar{y}_d$. To prove the second claim we define the tracking error to be $e = \xi - \bar{y}$. Then $\dot{e} = Ae + \bar{\psi}(\xi, \eta, v(\xi, \bar{y}_d))$. Now since we know that $\bar{\psi}$ contains only higher order terms and those terms are bounded by $K\varepsilon^2$ we conclude that the tracking error is $O(\varepsilon^2)$.

Remark 4.3: In [BI], using similar arguments, it was shown that if the system has regular zero dynamics and is minimum phase then it is stabilizable by the state feedback. Therefore this result is a counterpart of that result when the design objective is tracking.

REFERENCES

1. [BI] C.I. Byrnes, A. Isidori, *Local Stabilization of minimum phase nonlinear system*, Systems & Control Letters **11** (1989), 9-17.
2. [DBG] M.D. DiBenedetto, J.W. Grizze, *An Intrinsic Notion of Regularity for Local Output Nulling, Inversion and Dynamic Extension of Nonsquare Systems, preprint submitted to C-TAT.*
3. [DBGM] M.D. DiBenedetto, J.W. Grizzle, C.H. Moog, *Rank Invariants of Nonlinear Systems*, SIAM J. Control **72, No. 3** (1989), 658-672.
4. [GDBHS] J.W. Grizzle, M.D. DiBenedetto, J. Hauser, S. Sastry, *Approximation by Regular Input Output Maps*, preprint.
5. [H] W. Hahn, "Stability of Motion," Springer Verlag, Berlin, 1967.
6. [HSK] J. Hauser, S. Sastry, P. Kokotovic, *Nonlinear Control via Approximate Input Output Linearization: the Ball and Beam Example*, Proceedings of the 29th CDC, Tampa Fl (1989).
7. [I] A. Isidori, "Nonlinear Control Systems," 2nd ed., Springer Verlag, Berlin Heidelberg, 1989.
8. [K] A. J. Krener, *Approximate Linearization by State Feedback and coordinate change*, Systems and Control Letters **5** (1984), 181-185.

MINIMAX STATE ESTIMATION FOR SOME NONLINEAR STOCHASTIC SYSTEMS WITH UNCERTAIN STATISTICS

Engin Yaz
Electrical Engineering Department
University of Arkansas
Fayetteville, AR 72701

ABSTRACT

In this work, we present a minimax design of linear state estimators for a class of discrete-time nonlinear stochastic systems based on noise-corrupted nonlinear measurements.

INTRODUCTION

The design of linear minimax filters is considered for state estimation of a class of discrete-time nonlinear stochastic systems with noisy nonlinear measurement equations. First, the signal and measure- ment models are introduced. Then the linear minimum variance estimator is presented. We refer the reader to the works [1] - [4] for a general overview of the nonlinear estimation area. In this work, we will be concerned with minimax filter design for the specific class of nonlinear models that will be introduced, when there is second moment uncertainty in the noise statistics and necessary and sufficient conditions will be given for a saddle point. Similar approaches have been applied to linear models with additive noise [5] - [6] and multiplicative noise [7]. Our results will include the previous discrete-time results as special cases.

The following notation will be used: Z^+, $\mathbb{R}^n$, $\mathbb{R}^{nxm}$, S_n^+ will respectively denote the set of positive integers, n-dimensional real Euclidean space, the set of (nxm)-dimensional matrices and the cone of nonnegative definite symmetric (nxn)-dimensional matrices. x^T, $||x||$, tr[A], $\rho(A)$, $A\otimes B$, st(A) are respectively the transpose of x, vector 2-norm (Euclidean norm) of $x\epsilon\mathbb{R}^n$, trace of square matrix A, spectral radius of A, Kronecker product of A and B, and the stacking (vector) operator applied on matrix A. $\omega\epsilon\Omega$, where $\{\Omega, \mathcal{F}, P\}$ is a probability space, E{.} denotes expectation and E{x/y} the conditional expectation of x given on y.

MAIN RESULTS

We consider the following system and measurement nonlinear stochastic equations in discrete time:

$$x_{k+1} = A_k x_k + f_k(x_k, v_k(\omega)) \tag{1a}$$

$$y_k = C_k x_k + g_k(x_k, w_k(\omega)) \tag{1b}$$

where $x_k \epsilon \mathbb{R}^n$ is the state, $y_k \epsilon \mathbb{R}^m$ the output, $v_k \epsilon \mathbb{R}^p$ and $w_k \epsilon \mathbb{R}^q$ are the process and measurement noises respectively. These noises are assumed to be zero mean, independent and second order. The initial state which is independent of v_k and w_k, has mean $\bar{x}_o$ and covariance S_o. f_k: $Z^+ x\, \mathbb{R}^n x \mathbb{R}^p \rightarrow \mathbb{R}^n$ and g_k: $Z^+ x \mathbb{R}^n x \mathbb{R}^q \rightarrow \mathbb{R}^m$ have the following properties:

$$E\{f_k/x_k\}=0,\ E\{f_k f_k^T/x_k\} = P_{1k}^o + \sum_{i=1}^{r_1} P_{1k}^i\, x_k^T\, T_{1k}^i\, x_k \tag{2a}$$

$$E\{g_k/x_k\}=0,\ E\{g_k g_k^T/x_k\} = P_{2k}^o + \sum_{i=1}^{r_2} P_{2k}^i\, x_k^T\, T_{2k}^i\, x_k \tag{2b}$$

$$E\{f_k g_k^T/x_k\}=0 \tag{2c}$$

for all x_k, where r_1, $r_2 \epsilon Z^+$, P_{1k}^o, P_{1k}^i, T_{1k}^i, T_{2k}^i ϵS_n^+, P_{2k}^o, $P_{2k}^i \epsilon S_m^+$ for all i and let $P_{2k}^o > 0$.

This model permits us to handle several important classes of stochastic systems. For example, linear systems and measurement models with white additive noise $B_k v_k(w)$ which is, by far, the most common noise model used, systems or measurement models with white multiplicative noise $b_k^T x_k v_k(w)$, $b_k \epsilon \mathbb{R}^n$, or so called "bilinear stochastic systems" [8] - [10], systems or measurement models involving noise-corrupted nonlinear functions of the state: $\|x_k\|\ B_k v_k(\omega)$, $|b_k^T x_k|\ B_k v_k(\omega)$, or $\mathrm{sgn}[\phi(x_k)] B_k v_k(\omega)$ where sgn denotes the signum function, can all be included in this framework. The control of such systems has been considered in [11] - [13] and the linear minimum variance state estimation problem is solved in [14] and the existence, stability and robustness of the time invariant filter is discussed. In this paper, we will assume that the covariance matrices in (2a) - (2b) are not known and we will find necessary and sufficient conditions for the existence of a saddle point for a mean square estimator error criterion. That is, we find the estimator that minimizes the mean square estimator error assuming the worst possible values of the unknown covariances.

Define $\hat{x}_k$ as the estimate of x_k, $e_k = x_k - \hat{x}_k$ as the estimation error and the performance functional

$$J = e_N^T U e_N,\ U \epsilon S_n^+ \tag{3}$$

If we confine ourselves to linear estimators

$$\hat{x}_{k+1} = D_k\hat{x}_k + K_k y_k \tag{4}$$

the error dynamics will be

$$e_{k+1} = (A_k - K_k\ C_k)x_k + D_k\ \hat{x}_k + f_k - K_k g_k \tag{5}$$

Taking the expected value of both sides and imposing the unbiasedness condition gives the estimator

$$\hat{x}_{k+1} = A_k\hat{x}_k + K_k(y_k - C_k\ \hat{x}_k) \tag{6}$$

Therefore, the system and the estimation error evolve according to

$$\begin{bmatrix} x_{k+1} \\ \\ e_{k+1} \end{bmatrix} = \begin{bmatrix} A_k & 0 \\ \\ 0 & A_k - K_c C_k \end{bmatrix} \begin{bmatrix} x_k \\ \\ e_k \end{bmatrix} + \begin{bmatrix} f_k \\ \\ f_k - Kg_k \end{bmatrix} \overset{\Delta}{=} A_k^c \begin{bmatrix} x_k \\ e_k \end{bmatrix} + \mathfrak{F}_k \tag{7}$$

Introducing $X_k = \begin{bmatrix} X_k^{11} & X_k^{12} \\ (X_k^{12})^T & X_k^{22} \end{bmatrix} \overset{\Delta}{=} E\{ \begin{bmatrix} x_k \\ e_k \end{bmatrix} [x_k^T\ e_k^T]\}$, (2) and (7) give

$$X_{k+1} = A_k^c\ X_k\ A_k^{cT} + E\{\mathfrak{F}_k\mathfrak{F}_k^T\} \tag{8}$$

where

$$E\{\mathfrak{F}_k\mathfrak{F}_k^T\} = \begin{bmatrix} \Sigma_1(X_k^{11}) & \Sigma_1(X_k^{11}) \\ \Sigma_1(X_k^{11}) & \Sigma_1(X_k^{11}) + K_k\Sigma_2(X_k^{11})K_k^T \end{bmatrix}$$

with $\Sigma_1(X_k^{11}) = P_{1k}^o + \sum_{i=1}^{r_1} P_{1k}^i\ tr[X_k^{11}T_{1k}^i]$ and $\Sigma_2(X_k^{11}) = P_{2k}^o + \sum_{i=1}^{r_2} P_{2k}^i tr[X_k^{11}T_{2k}^i]$.

We rewrite (3) as

$$J = tr[HX_N] \tag{9}$$

where H = block diag[0, U]. At this point, we assume that the covariance matrices are known and minimize (9) subject to (8). The solution to this problem can be found by using matrix minimum principle [15] similar to the approach used in [14] and comes out to be as follows:

The optimal filter gain is given as:

$$K_k = A_kX^{22}_kC^T_k(C_kX^{22}_kC^T_k + \Sigma_2(X^{11}_k))^{-1} \tag{10}$$

which depends on the second moment of the state X^{11}_k which evolves as

$$X^{11}_{k+1} = A_kX^{11}_k A^T_k + \Sigma_1(X^{11}_k) \overset{\Delta}{=} \Phi_1(X^{11}_k) \tag{11}$$

and the minimum estimation error covariance X^{22}_k whose propagation is according to

$$X^{22}_{k+1}=A_kX^{22}_kA^T_k-A_kX^{22}_kC_k(C_kX^{22}_kC^T_k+\Sigma_2(X^{11}_k))^{-1}C^T_kX^{22}_kA^T_k+\Sigma_1(X^{11}_k)\overset{\Delta}{=}\Phi_2(X^{11}_k,X^{22}_k) \tag{12}$$

with $X^{12}_k = (X^{12}_k)^T = X^{22}_k$, $X^{11}_o = \bar{x}_o\bar{x}^T_o + S_o$, $X^{22}_o = S_o$

These conditions are both necessary and sufficient to have linear unbiased minimum variance estimation when the noise statistics are known. If they are not known, then they are assumed to belong to compact sets and we resort to the worst-case design procedure of minimizing the quadratic cost (9) based on the worst possible values of the noise covariances. That is, we will search for the linear filter that is optimal in the following sense

$$J_{mM} = \min_{K_k} \max_{\theta} J \tag{13}$$

where θ is the set of unknown noise covariances: $\theta = \{P^o_{1k},P^i_{1k},T^i_{1k},P^o_{2k},P^j_{2k},T^j_{2k},\ 1\le i\le r_1,\ 1\le j\le r_2\}$ subject to (8). If a saddle point exists, then the solution to (13) is the same as the solution to [16]:

$$J_{Mm} = \max_{\theta} \min_{K_k} J \tag{14}$$

The necessary and sufficient condition for the equivalence of (13) and (14) is

$$J(K^*_k,\ \theta) \le J(K^*_k,\ \theta^*) \le J(K_k,\ \theta^*) \tag{15}$$

By introducing the Hamiltonian $\mathcal{H}$ as

$$\mathcal{H} = \mathcal{H}(K_k,\ \theta) = \mathrm{tr}[X_{k+1}\ \ Y_{k+1}] \tag{16}$$

where $Y_k = \begin{bmatrix} Y_k^{11} & Y_k^{12} \\ (Y_k^{12})^T & Y_k^{22} \end{bmatrix}$ is the symmetric costate matrix, if (K_k^*, θ^*) is the saddle point, then

$$\mathcal{H}(K_k^*, \theta) \le \mathcal{H}(K_k^*, \theta^*) \le \mathcal{H}(K_k, \theta^*) \tag{17}$$

The main result on the neccessary and sufficient conditions for a point to be a saddle point is given as follows:

Theorem 1. (K_k^*, θ^*) is a saddle point for (13) if and only if

$$\mathrm{tr}[(\Sigma_1^*(X_k^{11}) - \Sigma_1(X_k^{11})(Y_{k+1}^{11} + Y_{k+1}^{22})] \ge 0 \tag{18}$$

$$\mathrm{tr}[K_k^*(\Sigma_2^*(X_k^{11}) - \Sigma_2(X_k^{11}))\, K_k^{*T}\, Y_{k+1}^{22}] \ge 0 \tag{19}$$

where $\Sigma_1^*(X_k^{11}) = (P_{1k}^o)^* + \sum_{i=1}^{r_1} (P_{1k}^i)^* \,\mathrm{tr}[X_k^{11}(T_{1k}^{i*})]$ and similarly for $\Sigma_2^*(X_k^{11})$.

Also

$$Y_k^{11} = A_k^T Y_{k+1}^{11} A_k + \sum_{i=1}^{r_1} \mathrm{tr}[P_{1k}^i (Y_{k+1}^{11} + Y_{k+1}^{22})] T_{1k}^i + \sum_{i=1}^{r_2} \mathrm{tr}[K_k^* P_{2k}^i K_k^{*T} Y_{k+1}^{22}] T_{2k}^i \overset{\Delta}{=} \Gamma_1(Y_{k+1}^{11}, Y_{k+1}^{22}),$$

$$Y_N^{11} = 0 \tag{20}$$

$$Y_k^{22} = (A_k - K_k^* C_k)^T\, Y_{k+1}^{22}(A_k - K_k^*\, C_k) \overset{\Delta}{=} \Gamma_2(Y_{k+1}^{22}),\ Y_N^{22} = U \tag{21}$$

with $Y_k^{12} = 0$.

Proof. (Necessity) Let $\theta = \theta^*$ in (16) to obtain (18) and (19). (Sufficiency) Suppose θ^* satisfies (18) and (19) with K^* given by (10) and Y_k given by (20)-(21). (18) and (19) will imply the left side inequality in (17) and the right side inequality results from the optimality of K^*.

Since the saddle point for J is the saddle point of $\mathcal{H}$, the problem can be divided into two sub-problems:

First, finding the minimum of J in (9) over K_k subject to (8), (18), and (19) which yields K^* in (10) where X_k^{11} and X_k^{22} given by (11) and (12) and all the quantities are evaluated at $\Theta = \Theta^*$.

Second, finding the maximum of J in (9) over Θ, subject to (8), (10) - (12), which yields (18) - (19) and $Y_k = \partial\mathcal{H}/\partial X_k$ gives (20) - (21). This concludes the proof.

Although the linear minimum variance filter is not a function of the costate Y_k or the matrix U in the performance functional, the minimax filter contains both variables.

If the set of unknown covariances Θ has maximal points such that $P^0_{1k} \le (P^0_{1k})^*$, $P^i_{1k} \le (P^i_{1k})^*$, etc., then these maximal elements will satisfy the conditions for a saddle point.

TIME-INVARIANT CASE

In case when A_k and C_k are replaced by constant matrices A and C and all the covariances on the right side of (20) - (26) are replaced by their constant counterparts, the performance functional is modified as

$$J = \lim_{N\to\infty} E\{e_N^T U e_N\} \tag{22}$$

The necessary and sufficient conditions for a saddle point follow trivially from the time-varying case and are given as:

Theorem 2. The point (K^*, Θ^*) is a saddle point if and only if the following conditions hold:

$$K^* = AX^{22}C^T(CX^{22}C^T + \Sigma_2(X^{11}))^{-1} \tag{23}$$

$$X^{11} = \Phi_1(X^{11}) \tag{24}$$

$$X^{22} = \Phi_2(X^{11}, X^{22}) \tag{25}$$

$$Y^{11} = \Gamma_1(Y^{11}, Y^{22}) \tag{26}$$

$$Y^{22} = \Gamma_2(Y^{22}) \tag{27}$$

$$\mathrm{tr}[(\Sigma_1^*(X^{11}) - \Sigma_1(X^{11}))\ (Y^{11} + Y^{22})] \ge 0 \tag{28}$$

$$\mathrm{tr}[K^*(\Sigma_2^*(X^{11}) - \Sigma_2(X^{11}))K^{*T}Y^{22}] \ge 0 \tag{29}$$

One can see from the above result that for the existence of the time invariant filter, it is necessary for the $\lim_{k\to\infty} X_k^{11}$ to exist. This requires the mean square boundedness of the system state, the necessary and sufficiency condition for which can be written as follows [14]:

$$\rho\{A \otimes A + \sum_{i=1}^{r_1} st(P_1^i)\ st^T(T_1^i)\} < 1 \tag{30}$$

A general existence result for a saddle point is given below:

Theorem 3. If (30) holds so that the stationary system (1) has a steady-state and the unknown covariances belong to compact and convex sets, then a saddle point solution always exists. Proof. From (1a) - (2b), unbiasedness condition and (5), we have

$$e_{k+1} = (A - KC)e_k + f_k - Kg_k \tag{31}$$

or

$$e_N = (A - KC)^N e_o + \sum_{k=0}^{N-1} (A - KC)^{N-k-1}(f_k - Kg_k) \tag{32}$$

which can be rearranged to be written in the form

$$e_N = F_N \zeta_N$$

where the matrix F_N represents the linear filter and $\zeta_N=[x_o^T, f_o^T, f_1^T, \ldots, g_o^T, g_1^T, \ldots]^T$. The covariance of the estimation error $E\{e_N e_N^T\} = F_N\ E\{\zeta_N \zeta_N^T\} F_N^T$ can easily be shown to be convex in F_N (in the class of filters) and concave in $E\{\zeta_N \zeta_N^T\}$ (in the class of unknown covariances), therefore a saddle point solution always exists [16].

REFERENCES

[1] A. P. Sage and J. L. Melsa, Estimation Theory with Applications to Communications and Control, McGraw Hill: New York, 1971.

[2] B. D. O. Anderson and J. B. Moore, Optimal Filtering, Englewood Cliffs N.J.: Prentice-Hall, 1979.

[3] S. I. Marcus, "Algebraic and geometric methods in nonlinear filtering," SIAM J. Contr. & Optim., vol. 22, pp. 817 - 844, 1984.

[4] V. Krishnan, Nonlinear Filtering & Smoothing: An Introduction to Martingales, Stochastic Integrals & Estimation, New York: Wiley, 1984.

[5] V. Poor and D. P. Looze, "Minimax state estimation for linear stochastic systems with noise uncertainty," IEEE Trans. Autom. Contr., vol. 26, pp. 902 - 906, 1981.

[6] G. A. Golubev, "Minimax linear filtering of dynamic discrete-time processes," Autom. Remote Contr., vol. 45, pp. 203-211, 1984.

[7] Y. A. Phillis, "Estimation and control of systems with unknown covariance and multiplicative noise," Proc. 26th CDC, Los Angeles, CA, pp. 321-325, 1987.

[8] R. R. Mohler and W. J. Kolodziej, "An overview of stochastic bilinear control processes," IEEE Trans. Syst. Man Cyber., vol. 10, pp. 913-919, 1980.

[9] W. L. DeKoning, "Optimal estimation of linear discrete-time systems with stochastic parameters," Automatica, vol. 20, pp. 113-115, 1984.

[10] E. Yaz, "Observer design for stochastic-parameter systems," Int. J. Contr., vol. 46, pp. 1213 - 1217, 1987.

[11] D. H. Jacobson, D. H. Martin, M. Pachter, and T. Geveci, Extensions of LQ Control Theory, Berlin: Springer-Verlag, 1980.

[12] E. Yaz, "A control scheme for a class of discrete nonlinear stochastic systems," IEEE Trans. Autom. Contr., vol. 32, pp. 77-80, 1987.

[13] E. Yaz, "Infinite horizon quadratic optimal control of a class of nonlinear stochastic systems," IEEE Trans. Autom. Contr., vol. 34, 1176 - 1180, 1989.

[14] E. Yaz, "Linear state estimators for non-linear stochastic systems with noisy non-linear observations," Int. J. Contr., vol. 48, pp. 2465-2475, 1988.

[15] M. Athans, "The matrix minimum principle," Inform & Contr., vol. 11, pp. 592 - 606, 1968.

[16] T. Basar and G. J. Olsder, Dynamic Noncooperative Game Theory, New York: Academic Press, 1982.

A FRANK-WOLFE / GRADIENT PROJECTION METHOD FOR SOLVING OPTIMAL CONTROL PROBLEMS

GERARD G. L. MEYER
Electrical and Computer Engineering Department
The Johns Hopkins University
Baltimore, Maryland 21218

LOUIS J. PODRAZIK
Supercomputing Research Center
Institute for Defense Analyses
Bowie, Maryland 20715-4300

Abstract

We propose a new parametrized gradient projection algorithm for solving large scale problems, and in particular discrete optimal control problems with linear constraints. We show that the Frank-Wolfe and Rosen gradient projection methods are both special cases of the proposed algorithm with the appropriate choice of algorithmic parameters. We investigate the identification of those algorithm parameters that result in fast convergence to the solution by allowing many constraints to be added or dropped from the active set at each iteration. As a consequence, the proposed algorithm overcomes the typically slow progress exhibited by classical active set projection methods on problems in which the solution lies on a large number of constraints. We further show that in the case of a quadratic objective, a Fletcher-Reeves type conjugate gradient modification for manifold suboptimization results in the algorithm converging to a nondegenerate solution point in a finite number of iterations. Numerical results run on the Cray2 and Sun4 are provided for discrete optimal control problems with a large number (≥ 1000) of control variables and compared against existing results.

1. Introduction

In this paper we present a parametrized gradient projection algorithm for solving large scale convex programming problems with linear inequality constraints and investigate the numerical behavior of the algorithm in the context of discrete optimal control problems. The proposed method has the capability to add and drop many constraints at each iteration, thereby overcoming the characteristically slow progress exhibited by classical projection methods in the case when many constraints are binding at the solution. Furthermore, the use of a conjugate gradient technique rather than a Newton-like approach to insure finite termination at a nondegenerate solution point in the case of quadratic cost results in an algorithm with a low computational and storage requirement.

The original gradient projection algorithm proposed by Rosen [ROS60] as well as other methods based upon reduced gradient, manifold suboptimization and active set strategies (see, for example [GIL74]) may be inefficient when used to solve large scale problems with many constraints binding at the solution. This is due to the fact that at most one constraint is added to the active set at each iteration, resulting in at least as many iterations as there are active constraints at the solution. Moreover, in order to preclude the possibility of jamming at non-optimal points, an ε procedure [POL71] can be used: at iteration k, a parameter ε^k is chosen and one minimizes over the direction obtained by projecting the gradient onto the tangent linear manifold defined by the set of ε-active constraints.

To overcome the problem of slow progress being made toward the solution of the problem when many constraints are active, the Goldstein-Levitin-Polyak [GOL64], [LEV66] class of gradient projection techniques was proposed. This "bent-leg" class of projection methods does not stop as soon as a new constraint is encountered; rather it continues along the boundary according to a step length α^k as

$$x^{k+1} = P_\Omega[x^k - \alpha^k \nabla f^0(x^k)],$$

where $x^k \in E^n$ is the current iterate, $f^0(.)$ is the continuously differentiable objective function, Ω is the convex constraint set and $P_\Omega[z]$ denotes the unique projection of a vector $z \in E^n$ on Ω. In general, exact one-dimensional minimization to determine α^k is impractical due to the requirement of minimizing a piecewise continuously differentiable function. Thus, one may either use a fixed α^k chosen as a function of the Lipschitz constant of the gradient or use a suboptimal α^k. In the case of suboptimal α^k, Bertsekas used a modified Armijo-type procedure to obtain a gradient projection method [BER76] and a projected Newton-like method [BER82], both well suited for large scale problems with many simple constraints. Calamai and More [CAL87] generalize the results of Bertsekas to obtain convergence results that are independent of the specific method, requiring only that the gradient projection method is used on selected iterations.

The "bent-leg" approach has the capability of adding or dropping more than one constraint at each iteration, but only because each iteration contains a number of substeps corresponding to the implementation of the step size rule on the piecewise continuous constraint boundary. Our interest in large scale problems has directed us to consider an approach in which many constraints could also be easily added or dropped at each iteration. One such approach already exists, namely the Frank-Wolfe method [FRA56]. Thus, in order to develop an algorithm with the capability to add or drop many constraints at each iteration, we parametrize the gradient projection method so that it behaves similarly to the Frank-Wolfe method for the appropriate choice of the parameters. We show that a good choice of parameters controls the behavior of the proposed algorithm between that of the Frank-Wolfe and Rosen methods and results in a smaller number of iterations compared to either method alone.

Finite convergence on quadratic, linearly constrained problems usually results in good convergence results when the cost function is twice continuously differentiable. Thus, Gill and Murray [GIL74], Wright [WRI89a] and others investigated the use of Hessian, approximate Hessian, projected Hessian, and various update types in the context of gradient projection methods to obtain finite convergence on constrained quadratic problems. Many of those approaches become cumbersome when used to solve large problems. Our ultimate goal is an algorithm that can be implemented on specialized high speed parallel computer architectures to perform time critical large scale optimization. Thus, we decided against using a Newton-type or projected variable metric method in order to avoid the manipulation of large matrices. Instead, we modified our basic parametrized algorithm by incorporating a projected Fletcher-Reeves conjugate gradient type update when the active constraint set remains constant, thereby achieving finite convergence on a quadratic.

Finally, we give numerical examples of solving discrete linear quadratic regulator optimal control problems with control bounds, show the effects of parameter selection and examine how the parameters may be adaptively chosen at each iteration.

2. Preliminaries

We consider the following problem:

2.1 Problem. Find a point x^* in Ω such that for every x in Ω

$$f^0(x^*) \leq f^0(x),$$

where

i. $f^0(.)$ is a continuously differentiable and convex map from E^n into E,

ii. Ω is a non-empty subset of E^n defined by

$$\Omega = \{ x \mid f^i(x) \le 0 \text{ for } i = 1, 2, \ldots, m \},$$

iii. the maps $f^1(.), f^2(.), \ldots, f^m(.)$ from E^n into E are affine, that is

$$f^i(x) = \langle g^i, x \rangle - h_i, \text{ for } i = 1, 2, \ldots, m ,$$

iv. the vectors g^i for $i \in I(x^*,0)$ are linearly independent, where $I(x, 0)$ is the index set defined by

$$I(x,0) = \{ i \mid f^i(x) \ge 0 \} ,$$

and

v. a point x^1 in E^n exists such that the set $\{x \in \Omega \mid f(x) \le f(x^1)\}$ is bounded.

It is known, for example see [POL71], that Problem 2.1 possesses at least one solution and that the following necessary conditions of optimality are also sufficient.

2.2 Optimality Conditions. A point $x^* \in \Omega$ is a solution to Problem 2.1 if and only if there exists a vector $\mu \in E^m$ with components $\mu_i \ge 0, i = 1, 2, \ldots, m$, such that

$$\nabla f^0(x^*)^t + \sum_{i=1}^{m} \mu_i \nabla f^i(x^*)^t = 0 , \tag{2.1}$$

$$\mu_i f^i(x^*) = 0, \text{ for } i = 1, 2, \ldots, m, \tag{2.2}$$

where we define $\nabla f(x)$ to be the row vector $\left[\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n}\right]$.

Given a point $x \in E^n$, a scalar λ, and a non-empty, closed and convex subset S of E^n, let $w(x,\lambda,S)$ be the projection of $x - \lambda\nabla f^0(x)^t$ onto the set S, that is, $w(x,\lambda,S)$ satisfies

$$\| (x - \lambda\nabla f^0(x)^t) - w(x,\lambda,S) \| = \min_y \{ \| (x - \lambda\nabla f^0(x)^t) - y \| \mid y \in S \} ,$$

given a point $x \in \Omega$ and the parameter $\tau \ge 0$, let the τ-active constraints index set $I(x,\tau)$ be defined as

$$I(x,\tau) = \{ i \mid f^i(x) \ge -\tau \} ,$$

and given a subset I of the set $\{1, 2, \ldots, m\}$, let the subsets $\Omega(I)$ and $\hat{\Omega}(I)$ of E^n be defined as

$$\Omega(I) = \begin{cases} \{ y \mid f^i(y) \le 0 \text{ for all } i \in I \} & \text{if } I \ne \phi , \\ E^n & \text{if } I = \phi , \end{cases}$$

$$\hat{\Omega}(I) = \begin{cases} \{ y \mid f^i(y) = 0 \text{ for all } i \in I \} & \text{if } I \ne \phi , \\ E^n & \text{if } I = \phi . \end{cases}$$

The assumptions of Problem 2.1 imply that the set $\Omega(I)$ is non-empty, closed and convex, and $w(x,\lambda,\Omega(I))$ is well defined for every subset I of $\{1, 2, \ldots, m\}$. The set $\hat{\Omega}(I)$ is also closed and convex, but may be empty, and therefore $w(x,\lambda,\hat{\Omega}(I))$ will only be well defined when $\hat{\Omega}(I) \ne \phi$. Using the notation just defined, we may rewrite the optimality conditions as:

2.3 Lemma. If a point $x^* \in \Omega$ is a solution of Problem 2.1, then

$$w(x^*,\lambda,\Omega(I(x^*,\tau))) = x^* \tag{2.3}$$

for all $\lambda \ge 0$ and $\tau \ge 0$. Conversely, if x^* in Ω, $\lambda > 0$ and $\tau \ge 0$ satisfy Eq. (2.3), then x^* is a solution of

Problem 2.1.

3. A Parametrized Gradient Projection Algorithm

The fixed point reformulation given by Eq. (2.3) suggests the iteration

$$x^{k+1} = x^k + \alpha^k(w(x^k,\lambda,\Omega(I(x^k,\tau))) - x^k),$$

where α^k is chosen so that x^{k+1} is not only in Ω but also minimizes $f^0(.)$. The parameters λ and τ need not remain constant, and therefore we obtain the following gradient projection algorithm that is parametrized by two sequences $\{\lambda^k\}$ and $\{\tau^k\}$, which serve at each iteration to scale the gradient and define the approximation set $\Omega(I(x^k,\tau^k))$ upon which the projection is performed, respectively.

3.1 Algorithm. Let $x^1 \in \Omega$, $\{\lambda^k\} \geq \overline{\lambda} > 0$, and $\{\tau^k\} \geq 0$ be given.

Step 0: Let $k = 1$.

Step 1: Compute $p^k = w(x^k,\lambda^k,\Omega(I(x^k,\tau^k))) - x^k$.

Step 2: If $p^k = 0$ stop; else go to Step 3.

Step 3: Let $d^k = p^k$.

Step 4: Compute $\alpha^k \geq 0$ to minimize $f^0(x^k + \alpha^k d^k)$ subject to $(x^k + \alpha^k d^k) \in \Omega$.

Step 5: Let $x^{k+1} = x^k + \alpha^k d^k$.

Step 6: Let $k = k + 1$ and go to Step 1.

At each iteration, Algorithm 3.1 generates a search direction d^k which is computed by projecting $x^k - \lambda^k \nabla f^0(x^k)^t$ onto the set $\Omega(I(x^k,\tau^k))$. The parameter τ^k defines the "sufficiently-active" constraint region at iteration k, while λ scales the gradient. Thus, for τ^k large, Algorithm 3.1 reduces to projecting $x^k - \lambda^k \nabla f^0(x^k)^t$ onto the entire set Ω, and for $\tau^k = 0$, $\Omega(I(x^k,\tau^k)) = E^n$ whenever $x^k \in \text{interior}(\Omega)$, and Algorithm 3.1 reduces to a projected version of best-step steepest descent in which the direction chosen at an interior point is the negative gradient.

3.2 Remark. If $\tau^k = 0$ for $k \geq 1$, then Algorithm 3.1 is a version of the Rosen gradient projection method in which the direction obtained is just the steepest descent direction at x^k subject to the active inequality constraints.

3.3 Remark. λ^k must be bounded away from 0 to insure that Algorithm 3.1 possesses the appropriate convergence properties.

3.4 Remark. The quantity p^k is not required in the presentation of Algorithm 3.1, but is introduced to facilitate the modification of Algorithm 3.1 in subsequent sections .

We assume that the initial point x^1 used by Algorithm 3.1 satisfies part (v) of Problem 2.1, and therefore, if $\{x^k\}$ is a sequence generated by Algorithm 3.1, then $\{x^k\}$ is bounded, $f^0(x^{k+1}) < f^0(x^k)$ whenever x^{k+1} and x^k are two consecutive iterates, and using the standard techniques for analyzing the convergence of algorithms [POL71], it may be shown that:

3.5 Theorem. If $\{x^k\}$ is a sequence constructed by Algorithm 3.1 with $\tau^k > 0$ to solve Problem 2.1, then either $\{x^k\}$ is finite and the last point is optimal or $\{x^k\}$ is infinite and contains cluster points, each of which is optimal. Furthermore, when $f^0(.)$ is strictly convex, Problem 2.1 has a unique optimal solution x^*, and in that case $\{x^k\}$ converges to x^*.

It is interesting to note that Algorithm 3.1 belongs to the family of descent interior methods described in [MEY74], in which Step 1 of Algorithm 3.1 is replaced by the following:

3.6 Generalized p^k. Let $p^k = s^k - x^k$, where $s^k \in \Omega$ is such that $g(s^k) \leq g(s)$ for all $s \in \Omega$, the map $g(.)$ is defined by

$$g(s) = \nabla f^0(x^k)(s^k - x^k) + \frac{1}{2\lambda^k}(s - x^k)' B^k (s - x^k),$$

and B^k is a positive semi-definite $n \times n$ matrix.

In [MEY74] it was shown that if $B^k = 0$ and $\lambda^k = 1$, then Algorithm 3.1 is the algorithm proposed by M. Frank and P. Wolfe in 1956 [FRA56]. If B^k is the identity $n \times n$ matrix, and if $\lambda^k = 1$, then Algorithm 3.1 is the procedure proposed by J. B. Rosen in 1960 [ROS60]. If the matrix B^k is the Hessian of $f^0(.)$, or an approximation of the Hessian of $f^0(.)$, then Algorithm 3.1 is a second order or variable metric method. Each of these methods has advantages depending upon the structure of problem to be solved and our strategy in this paper consists of trying to identify the situation at hand; that is the type of problem that one is trying to solve, and then in choosing the parameters of Algorithm 3.1 in order to control the influence of each method so as to minimize the number of iterations necessary to obtain a satisfactory approximation of the solution.

4. Frank-Wolfe Algorithm

The Frank-Wolfe algorithm [FRA56] was proposed to solve the following problem:

4.1 Problem. Given a continuously differentiable and convex map $f^0(.)$ from E^n into E, and a non-empty, closed, bounded and convex subset Ω of E^n, find a point x^* in Ω such that for every x in Ω

$$f^0(x^*) \leq f^0(x).$$

At every iteration, the Frank-Wolfe algorithm generates a direction by minimizing the appropriate linear approximation of $f^0(.)$ over the set Ω, and then performs a line search along that direction, that is, the Frank-Wolfe algorithm is identical to Algorithm 3.1, except that the computation of p^k in Step 1 of Algorithm 3.1 is replaced by the following:

4.2 Frank-Wolfe p^k. Compute $s^k \in \Omega$ such that $\nabla f^0(x^k)s^k \leq \nabla f^0(x^k)s$ for all $s \in \Omega$, and let $p^k = s^k - x^k$.

It is known [CAN68] that the Frank-Wolfe algorithm may generate sequences that approach a solution x^* of Problem 4.1 rather slowly: given $\varepsilon > 0$ and $\alpha > 0$, $f^0(x^k) - f^0(x^*) \geq \alpha / k^{1+\varepsilon}$ for an infinite number of k. However, it is also true that the Frank-Wolfe algorithm exhibits finite convergence whenever the hypothesis below is satisfied.

4.3 Hypothesis. If x^* is a solution of Problem 4.1, a scalar $\varepsilon > 0$ exists such that

$$\nabla f^0(x)x^* \leq \nabla f^0(x)s,$$

for all $x \in B(x^*, \varepsilon) \cap \Omega$, and for all $s \in \Omega$.

Hypothesis 4.3 insures that if an iterate x^k is "close enough" to a solution of Problem 4.1, then the Frank-Wolfe direction p^k is such that the segment $[x^k, x^k + p^k]$ contains a solution of Problem 4.1.

When λ^k and r^k are large, Algorithm 3.1 behaves like the Frank-Wolfe algorithm. It is clear that if Hypothesis 4.3 is satisfied, and Ω is bounded, then Algorithm 3.1 generates a solution of Problem 2.1 in a finite number of iterations whenever λ^k and r^k are large enough. It follows that a possible strategy for choosing the sequences $\{\lambda^k\}$ and $\{r^k\}$ will consist of using large values for λ^k and r^k at least for the initial iterates in order to quickly approach the neighborhood of the solution if it is the case where a large number of constraints are binding at the optimum value.

5. Fletcher-Reeves Algorithm

We want Algorithm 3.1 to be efficient, and to that effect we add a Fletcher-Reeves [FLE64] modification to the algorithm. In order to make the modification as clear as possible, we repeat the Fletcher-Reeves procedure given in [GIL73, p.134] to solve the following variation of Problem 2.1:

5.1 Problem. Given a subset I of $\{1, 2, ..., m\}$ such that $\hat{\Omega}(I) \neq \phi$, find a point x^* in the linear manifold $\hat{\Omega}(I)$ such that for every x in $\hat{\Omega}(I)$

$$f^0(x^*) \leq f^0(x).$$

The Fletcher-Reeves algorithm [FLE64] consists of modifying the best step steepest descent method so that the sequence of directions used by the algorithm are conjugate whenever the map $f^0(.)$ is a quadratic. In order to insure global convergence when $f^0(.)$ is only continuously differentiable, one uses a spacer step every $\bar{k}$ iterations, that is, one uses a projected best step steepest descent algorithm every $\bar{k}$ iterations. Note that in the context of the Fletcher-Reeves algorithm, using a spacer step is also called restarting the algorithm.

5.2 Algorithm. Let x^1 in $\hat{\Omega}(I)$, $\lambda > 0$, and $\bar{k} \geq 0$ be given.

Step 0: Let $k = 1, c = 0$, and $\beta^0 = 0$.

Step 1: Compute $\hat{p}^k = w(x^k, \lambda, \hat{\Omega}(I)) - x^k$.

Step 2: If $\hat{p}^k = 0$ then stop; else go to Step 3.

Step 3: If $\beta^{k-1} = 1$ and if $c \leq \bar{k}$, let $c = c + 1$, let $d^k = \hat{p}^k + \dfrac{\|\hat{p}^k\|^2}{\|\hat{p}^{k-1}\|^2} d^{k-1}$, and got to Step 5; else go to Step 4.

Step 4: Let $\beta^k = 1$, let $c = 0$, and let $d^k = \hat{p}^k$.

Step 5: Compute $\alpha^k \geq 0$ to minimize $f^0(x^k + \alpha^k d^k)$.

Step 6: Let $x^{k+1} = x^k + \alpha^k d^k$.

Step 7: Let $k = k + 1$ and go to Step 1.

The quantity β^k is used to keep track of the nature of $\hat{p}^k$ and will play a bigger role later when the Fletcher-Reeves algorithm will be combined with the parametrized gradient projection algorithm. In the context of the Fletcher-Reeves algorithm, the only time when $\beta^k = 0$ is for $k = 0$.

If $\bar{k} = 0$, Algorithm 5.2 is the steepest descent algorithm restricted to $\hat{\Omega}(I)$, and if $\bar{k} > 0$, Algorithm 5.2 is a partial conjugate gradient method restricted to $\hat{\Omega}(I)$. Thus, if $\bar{k} \geq n - \|I\|$, and if $f^0(.)$ is a quadratic, Algorithm 5.2 will find the solution of Problem 5.1 in $n - \|I\|$ iterations.

6. Parametrized Fletcher-Reeves Gradient Projection Algorithm

We now combine Algorithms 3.1 and 5.2 to produce a parametrized gradient projection algorithm with the Fletcher-Reeves modification for solving Problem 2.1.

6.1 Algorithm. Let $x^1 \in \Omega$, $\{\lambda^k\} \geq \bar{\lambda} > 0$, $\{r^k\} \geq 0$, and $\{\psi^k\}$ in $[0,1]$ be given.

```
Step 0: Let $k = 1, c = 1, x^0 = x^1$, and $\beta^0 = 0$.
Step 1: Compute $p^k = w(x^k, \lambda^k, \Omega(I(x^k, r^k))) - x^k$.
        Compute $\hat{p}^k = w(x^k, \lambda^k, \hat{\Omega}(I(x^k, 0))) - x^k$.
Step 2: If $p^k = 0$ stop; else go to Step 3.
Step 3: If $\hat{p}^k \neq 0$, $<p^k, \hat{p}^k> \geq (1-\psi^k) \|p^k\| \|\hat{p}^k\|$, and $I(x^k,0) = I(x^{k-1},0)$, let $\beta^k = 1$; else let $\beta^k = 0$.
Step 4: If $\beta^k = 0$, let $c = 1$, let $d^k = p^k$, and go to Step 7; else go to Step 5.
Step 5: If $\beta^{k-1} = 1$ and $c < n - \|I(x^k,0)\|$, let $c = c + 1$, let $d^k = \hat{p}^k + \frac{\|\hat{p}^k\|^2}{\|\hat{p}^{k-1}\|^2} d^{k-1}$, and go to Step 7; else go to Step 6.
Step 6: Let $c = 1$, and let $d^k = \hat{p}^k$.
Step 7: Compute $\alpha^k \geq 0$ to minimize $f^0(x^k + \alpha^k d^k)$ subject to $(x^k + \alpha^k d^k) \in \Omega$.
Step 7: Let $x^{k+1} = x^k + \alpha^k d^k$.
Step 8: Let $k = k + 1$ and go to Step 1.
```

6.2 Remark. By assumption the vectors g^i, $i \in I(x, 0)$ are linearly independent. It follows that the set $\hat{\Omega}(I(x^k,0))$ is non-empty, and therefore the quantity $\hat{p}^k$ is well defined for every index $k \geq 1$.

6.3 Remark. We may have $p^k \neq 0$ and $\hat{p}^k = 0$, and therefore we must check the magnitude of $\hat{p}^k$ before checking the cosine of the angle between p^k and $\hat{p}^k$ in Step 3.

The quantity β^k is generated so that one can decide whether to use the Fletcher-Reeves update or not. If $p^k \neq 0$, $\hat{p}^k \neq 0$, if the angle between the vectors p^k and $\hat{p}^k$ is small enough as specified by the parameter ψ^k, and the active constraint set has not changed from iteration k-1, that is $I(x^k,0) = I(x^{k-1},0)$, then β^k is set equal to 1, otherwise β^k is set equal to 0. When $\beta^k = 0$, we use p^k as the direction in which to proceed (this is the only mechanism that the algorithm can use to leave the current linear manifold $\hat{\Omega}(I(x^k,0))$. When $\beta^k = 1$, we use $\hat{p}^k$, the projection of the gradient on the linear manifold defined by the active constraints as the direction in which to proceed whenever either $\beta^{k-1} = 0$, or we have used the Fletcher-Reeves update often enough, that is the count c is larger than $n - \|I(x^k,0)\|$. When $\beta^k = \beta^{k-1} = 1$, $I(x^k,0) = I(x^{k-1},0)$, and the count c is small enough, we use the Fletcher-Reeves update.

The reason for using the parameter sequence $\{\psi^k\}$ is to allow the use of a Fletcher-Reeves update even when p^k and $\hat{p}^k$ are not exactly collinear, but make a small angle. In that case, we can achieve better convergence results when the problem to be solved involves a function $f^0(.)$ which has an unconstrained minimum close to the boundary of Ω.

6.4 Remark. If $\psi^k < 0$, then $<p^k, \hat{p}^k> < (1 - \psi^k) \|p^k\| \|\hat{p}^k\|$, $\beta^k = 0$, $c = 1$, and $d^k = p^k$. It follows that if $\psi^k < 0$ for all k, then Algorithm 6.1 reduces to Algorithm 3.1.

6.5 Remark. When $f^0(.)$ is a quadratic, Algorithm 6.1 generates a new direction d^k which is conjugate to d^{k-1} whenever the point x^k is in the same tangent linear manifold passing through x^{k-1}, $\hat{\Omega}(I(x^{k-1},0))$ and the angle between p^k and $\hat{p}^k$ is small enough. However, since p^k is obtained by projecting onto the set $\Omega(I(x^k,r^k))$ rather than $\hat{\Omega}(I(x^k,0))$, Algorithm 6.1 is not a pure active set method.

6.6 Theorem. If $\{ x^k \}$ is a sequence constructed by Algorithm 6.1 with $\tau^k > 0$ to solve Problem 2.1 then either $\{ x^k \}$ is finite and the last point is optimal or $\{ x^k \}$ is infinite and contains cluster points, each of which is optimal. Furthermore, when a Problem 2.1 is such that $f^0(.)$ is strictly convex, then that problem has a unique optimal solution x^*, and in that case $\{ x^k \}$ converges to x^*.

We conclude this section by presenting the finite convergence properties of Algorithm 6.1 in the case of quadratic cost.

6.7 Definition. A stationary point $x^* \in \Omega$ is *nondegenerate* if the corresponding Lagrange multipliers $\mu_i^* > 0$ for all $i \in I(x^*,0)$.

6.8 Theorem. If $\{ x^k \}$ is a sequence constructed by Algorithm 6.1 with $\tau^k > 0$ to solve Problem 2.1 with quadratic cost and positive definite Hessian, then $\{ x^k \}$ converges to the nondegenerate solution x^* in a finite number of steps.

7. Numerical Results

We now give numerical results obtained in solving the following discrete optimal control problem [BER82]:

7.1 Problem.

$$\underset{u}{\text{minimize}}\, J(u) = \frac{1}{2} \sum_{i=1}^{N} \left(z_i^t \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} z_i + u_i^t \begin{bmatrix} 6 \end{bmatrix} u_i \right)$$

$$\text{subject to } z_i = \begin{bmatrix} 1.0 & s \\ -s & 1.0 \end{bmatrix} z_{i-1} + \begin{bmatrix} 0 \\ s \end{bmatrix} u_i, \quad i = 1, 2, \ldots, N,$$

$$u \in \Omega,$$

where $z_i \in E^2$ and $u_i \in E^1$, the convex compact set Ω is defined as

$$\Omega = \{ u \mid \|u_i\|_\infty \leq 1, \quad \text{for } i = 1, 2, \ldots, N \}.$$

The various cases considered use different choices of s as given in Tables 1 and 2 for the starting point $u^1 = 0$ and initial state $z_0 = (15, 5)$. Actual numerical results of Algorithm 6.1 run on a single head of the Cray2 and Sun4 are compared against the projected Newton methods implemented by Bertsekas [BER82] (machine unknown) and Wright [WRI89b] for a single processor on the Alliant FX/8. Stop rule precision was $\|p^k\|^2 \leq 10^{-10}$.

In Table 1 we compare the performance of Algorithm 6.1 in the case in which the parameters are held constant for every iteration with the values of $\tau^k = 1.0$, $\lambda^k = 1.0$ and $\psi^k = 0.1$. In Table 2, λ^k and ψ^k are again held fixed at $\lambda^k = 1.0$ and $\psi^k = 0.1$. However, $\tau^1 = 2.0$ and τ^k is adaptively changed at each iteration according to the rule

$$\tau^k = \tau^k * 2, \qquad \text{if } I(x^k,0) \neq \text{if } I(x^{k-1},0),$$

$$\tau^k = \tau^k / 2, \qquad \text{if } I(x^k,0) = \text{if } I(x^{k-1},0),$$

thus making Algorithm 6.1 generate directions similar to Frank-Wolfe and then tend toward the usual gradient projection directions as the iteration progresses and arrives at the final constraint manifold.

7.2 Numerical Results.

Table 1. Numerical Results for Problem 7.1 with Fixed $\tau^k = 1.0, \lambda^k = 1.0, \psi^k = 0.1$

N	s	# of binding constraints at solution	Algorithm [WRI89b][1]		Algorithm 6.1		
			# of iterations	time (Alliant/FX8)	# of iterations	time (Cray2)	time (Sun4)
1000	0.001	183	3	2.59s	9	0.201s	1.09s

Table 2. Numerical Results for Problem 7.1 with Adaptive τ^k, $\lambda^k = 1.0, \psi^k = 0.15$

N	s	# of binding constraints at solution	Algorithm [BER82][2]	Algorithm 6.1		
			# of iterations	# of iterations	time (Cray2)	time (Sun4)
100	0.002	0	1	2	0.00579s	0.027s
	0.01	18	3	8	0.0227s	0.103s
	0.04	78	3	5	0.0140s	0.073s
	0.1	91	4	5	0.0140s	0.059s
	0.5	100	5	3	0.0084s	0.036s
1000	0.0002	0	1	2	0.055s	0.281s
	0.001	183	3	8	0.221s	0.98s
	0.004	770	4	7	0.188s	1.50s
	0.01	890	4	19	0.504s	2.19s
	0.05	705	23	186	5.03s	28.71s
10000	0.00002	0	1	2	0.542s	3.01s
	0.0001	1834	11	9	2.92s	15.95s
	0.0004	7693	5	38	9.83s	5.18s
	0.001	8861	6	14	3.62s	17.27s
	0.005	4261	10	450	180.02s	595.03s

8. Conclusions

The numerical results given in Tables 1 and 2 indicate that the proposed parametrized projected conjugate gradient algorithm is competitive to the second order methods proposed in [BER82] and [WRI89b] in terms of both iteration count and processor execution time. Furthermore, the highly structured and simple computations required at each iteration suggest that the proposed method is suitable for high speed SIMD or MIMD parallel computer implementation.

---

[1] Iteration and timing data published for only one problem.

[2] Timing data not published.

9. References

[BER76] Bertsekas, D.P., On the Goldstein-Levitin-Polyak Gradient Projection Method, *IEEE Trans. Automatic Control,* Vol. AC-21 No. 2, April 1976, pp. 174-184.

[BER82] Bertsekas, D.P., Projected Newton Methods for Optimization Problems with Simple Constraints, *SIAM J. Control and Optimization,* Vol. 20, No. 2, March 1982, pp. 221-246.

[CAL87] Calamai, P.H. and More, J.J., Projected Methods for Linearly Constrained Problems, *Mathematical Programming 39*, North-Holland, 1987, pp. 99-116.

[CAN68] Canon, M. D., and Cullum, C. D., A Tight Upper Bound on the Rate of Convergence of the Frank-Wolfe Algorithm, *SIAM J. Control 6,* 1968, pp.509-516.

[FLE64] Fletcher, R., and Reeves, C.M., Function Minimization by Conjugate Gradients, *Computer J. 7*, 1964, pp 149-154.

[FRA56] Frank, M., and Wolfe, P., An Algorithm for Quadratic Programming, *Naval Res. Logist. Quart. 3,* 1956, pp. 95-110.

[GOL64] Goldstein, A.A., Convex Programming in Hilbert Space, *Bull. Amer. Math. Soc.,* 70, 1964, pp. 709-710.

[GIL74] Gill, P.E. and Murray, M., Numerical Methods for Constrained Optimization, *Academic Press*, New York, Chs. 2-4, 1974.

[LEV66] Levitin, E.S. and Polyak, Constrained Minimization Problems, *U.S.S.R. Comput. Math. Phys.*, 6, 1966, pp 1-50.

[MEY74] Meyer, G.G.L., Nonwastefulness of Interior Iterative Procedures, *Journal of Mathematical Analysis and Applications*, Vol. 45, No. 2, February, 1974, pp. 485-496.

[POL71] Polak, E.J., *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, New York, 1971.

[ROS60] Rosen, J.B., The Gradient Projection Method for Nonlinear Programming, Part I: Linear Constraints, *J. SIAM Control* Vol. 8, 1960, pp. 181-217.

[WOL67] Wolfe, P., *Methods of Nonlinear Programming,* J.Abadie (editor), Interscience, John Wiley and Sons, New York, New York, 1967, pp. 97-131.

[WRI89a] Wright, S.J., Convergence of SQP-Like Methods for Constrained Optimization, *SIAM J. Control and Optimization*, Vol. 27, No. 1, January 1989, pp. 13-26.

[WRI89b] Wright, S.J., Solution of Discrete-Time Optimal Control Problems on Parallel Computers, *Argonne National Laboratory Report MCS-P89-0789*, July, 1989.

CONTROLLED INVARIANCE FOR SINGULAR DISTRIBUTIONS

Viswanath Ramakrishna

Department of Systems Science and Mathematics

Washington University

St. Louis, Missouri 63130

Abstract: The problem of rendering an analytic distribution controlled invariant is considered. The solution is related to scaling and translational symmetries.

We consider here the problem of rendering an analytic distribution Δ controlled invariant for an analytic affine control system $\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i$. It is assumed

a) Δ is weakly (f, g) invariant, i.e.

$$[f, X_i] = V_i + \sum_{j=1}^{m} c_j^i g_j$$

$$\text{and } [g_j, X_i] = W_i + \sum_{k=1}^{m} b_{jk}^i g_k. \quad i = 1, \dots, d \;\; j = 1, \dots, m$$

for a local basis $\{X_i\}_{i=1,\dots,d}$ of Δ. Here V_i, W_i belong to Δ; and the (c_j^i) and b_{jk}^i are analytic functions. b) $(G + \Delta)/\Delta$ has constant dimension. Here $G = \text{Span}\{g_1, \dots, g_m\}$.

If we assume that Δ is input insensitive, then the required feedback may be found by solving the following system of PDE's:

$$I. \;\; L_{X_i}(\alpha_j) = c_j^i.$$

The integrability conditions for Δ are provided by a) and b). If, in addition Δ is nonsingular then (I) has solutions.

We now assume that Δ has singularities at the origin. Then one can find a basis for Δ consisting of coordinate vector fields and degenerate (i.e. vanishing at the origin) vector fields on a lower dimensional space. This shows that (I) entails solving a degenerate system of PDE's. In fact this suffices.

There are now 2 cases to consider: 1) Δ has a basis consisting entirely of degenerate vector fields, ii) there are coordinate vector fields in the basis for Δ.

In both cases we assume: iii) one of the degenerate vector fields is a LS vector field (i.e. has linear part scaling), iv) $f(0) = 0$.

Now, in case i) these assumptions (modulo a technicality) suffice to provide a solution. In case ii), it seems difficult to say more, if there is more than one degenerate PDE to deal with. However, using some techniques of symmetry [3], we are able to provide a qualified answer in this case. See [1] for a similar problem. See [2], and the references therein for the controlled invariance problem in general.

References

[1] C. I. Byrnes, *Feedback decoupling of rotational disturbances for spherically constrained systems,* Proceedings of the 23rd Conference on Decision and Control, Dec. 1984), Las Vegas, Nevada, pp. 421-426.

[2] A. Isidori, *Nonlinear Control Systems–An Introduction,* Springer-Verlag, 1983.

[3] V. Ramakrishna and H. Schättler, *Controlled Invariance and Group Invariance,* Submitted to Mathematical Systems and Control, May 1990.

EXACT COMPUTATION OF THE INFIMUM IN H$^\infty$-OPTIMIZATION VIA STATE FEEDBACK

Ben M. Chen[⋆]
Department of Electrical
and Computer Engineering
Washington State University
Pullman, WA 99164-2752

Ali Saberi[⋆]
Department of Electrical
and Computer Engineering
Washington State University
Pullman, WA 99164-2752

Uy-Loi Ly[⋆⋆]
Department of Aeronautics
and Astronautics, FS-10
University of Washington
Seattle, WA 98195

ABSTRACT

This paper presents a simple, accurate and non-iterative procedure for the computation of the exact value of the infimum in the standard H$^\infty$optimal control with state feedback. The problem formulation is general and does not place any restrictions on the direct feedthrough term between the control input and the controlled output variables. The algorithm involves solutions of two algebraic Lyapunov equations of a subsystem obtained from the transformation of the original system into a special coordinate basis. The method is applicable to a plant with no invariant zeros on the imaginary axis, and the transfer function matrix between the controls and the controlled outputs is right-invertible. The results are applied to the determination of the achievable reduction in H$^\infty$-norm of aircraft responses to turbulence in a disturbance rejection design using optimal state-feedback, and of the achievable H$^\infty$-performance in control of a flexible mechanical system.

1. INTRODUCTION

Over the past decade one has witnessed a proliferation of literature on H$^\infty$-optimal control since it was first introduced by Zames [1]. The main focus of the work has been and continues to be on the formulation and solution to the robust multivariable control problem. Since the original formulation in [1], a great deal of work has been on solving the H$^\infty$optimal control problem. Primarily all early research results involved a mixture of time-domain and frequency-domain techniques [2,3,4]. Recently, considerable attention has been focused on purely time-domain methods based on algebraic Riccati equations (ARE) [5-13]. Along this line of research, connections are also made between H$^\infty$-optimal control and differential games [14]. Typically in approaches to H$^\infty$-optimal control problems using algebraic Riccati solutions, the existence of suboptimal state- and output-feedback laws is formulated in terms of the existence of positive definite solutions for one or two indefinite algebraic matrix equations. The design solution is suboptimal in the sense that the H$^\infty$-norm of the closed-loop system transfer function from the disturbance to the controlled output is less than a prescribed value. A recent paper by Stoorvogel and Trentelman [11] has shown that conditions for the existence of suboptimal state-feedback laws can be expressed in terms

---

[⋆] The work of B. M. Chen and A. Saberi is supported in part by Boeing Commercial Airplane Group.

[⋆⋆] The work of U. Ly is supported in part by NASA Ames Dryden Flight Research Facility under grant contract NAG-2-629.

of the existence of a solution to a quadratic matrix inequality. Their conditions are very intuitive and reminiscent of the dissipation inequality in singular linear quadratic optimal control. Similar results are also obtained by Stoorvogel [10] for suboptimal output-feedback laws.

The state-space approach using algebraic Riccati solutions provides basically an iterative scheme of approximating the infimum (denoted here by γ_s^*) of the norm of the closed-loop transfer function under state-feedback laws. The procedure is as follows: one starts with a value of γ and then determines whether $\gamma > \gamma_s^*$ by solving an indefinite algebraic Riccati equation and checking the positive definiteness of the solution. In the case where such a positive definite solution exists then we have $\gamma > \gamma_s^*$, and the procedure is then repeated with a smaller value of γ . In principle, one can approximate the infimum γ_s^* to within any degree of accuracy in this manner. However this search procedure is exhaustive and can be very costly. More significantly due to the high-gain occurrence as γ gets close to γ_s^* , numerical solutions for algebraic Riccati equation can become highly sensitive and ill-conditioned. In fact this phenomenon can even occur when γ is not close to γ_s^* . So the iterative procedure based on algebraic Riccati solutions is not reliable and should not be used to determine the infimum γ_s^* . This paper presents a simple, accurate and non-iterative method of computing the exact value of γ_s^* .

In the computation of γ_s^* our problem formulation does not place any restrictions on the direct feedthrough matrix between the control input and the controlled output variables, removing the limitations imposed by Petersen [9] that require the existence of a nonsingular feedthrough term. One of the key components of our method is to transform the problem using a special coordinate basis (s.c.b) transformation introduced in [15], which exhibits clearly the finite- and infinite-zero structures of the system among other system geometric properties. The other component utilizes the results of Stoorvogel and Trentelman [11]. The algorithm for computing γ_s^* has been implemented in a Matlab-software environment. Numerous examples are given in section 5 to illustrate the computation of γ_s^* for aircraft control applications and control of a flexible mechanical system.

The outline of this paper is as follows. In section 2 we introduce the problem statement. In section 3 we recall the special coordinate basis (s.c.b) and its properties for non-strictly proper systems. This s.c.b transformation is instrumental in the derivation of the main results given in section 4 for the exact computation of γ_s^*. Section 5 contains numerous illustrative examples and the conclusion is given in section 6.

Throughout this paper we shall adopt the following conventions and notations:

A' := transpose of A.

I := an identity matrix of appropriate dimension.

$\Re$:= real numbers.

$\mathcal{C}$:= whole complex plane.

$\mathcal{C}^-$:= open left-half complex plane.

$\mathcal{C}^+$:= open right-half complex plane.

$\mathcal{C}^o$:= imaginary axis $j\omega$.

$\sigma_{max}(A)$:= maximum singular value of A.

$\lambda(A)$:= the set of eigenvalues of A.

We refer to the linear dynamical system

$$\dot{x} = Ax + Bu, \quad y = Cx + Du \tag{1}$$

as the system (A, B, C, D). We also refer to $T_{yu}(s) = C(sI - A)^{-1}B + D$ as the transfer function matrix of the system (A, B, C, D) between the input u and the output y. For any real rational matrix $T(s)$,

$$||T(s)||_\infty := \sup \{\sigma_{max}[T(j\omega)] : \omega \in \Re\} \tag{2}$$

then $||T(s)||_\infty$ coincides with the L_∞-norm of $T(s)$ if $T(s)$ is proper and has no poles in $\mathcal{C}^o$, and with the H^∞-norm of $T(s)$ if it is proper and stable.

2. PROBLEM FORMULATION

Let us consider the following linear system,

$$\dot{x} = Ax + Bu + Ew, \tag{3.a}$$

$$y = x, \tag{3.b}$$

$$z = Cx + Du, \tag{3.c}$$

where $x \in \Re^n$ is the state, $u \in \Re^m$ is the input, $w \in \Re^p$ is the disturbance, $y \in \Re^n$ is the measured output for feedback control (here we consider state feedback) and $z \in \Re^q$ is the controlled output. Let $T_{zw}(s)$ denote the transfer function matrix from the disturbance w to the controlled output z.

The standard H^∞optimal control problem with state feedback is concerned with the construction of internally stabilizing state-feedback control laws $u = Fx$ that minimize the H^∞norm of $T_{zw}(s)$, where

$$T_{zw}(s) = (C + DF)(sI - A - BF)^{-1}E.$$

We define

$$\gamma_s^* := \inf \{||T_{zw}(s)||_\infty \text{ where } u = Fx \text{ and } A + BF \text{ is a stability matrix}\}$$

to be the infimum of the H^∞-optimization under state-feedback laws. Most current state-space H^∞-optimization algorithms cannot determine γ_s^* exactly and can only provide lower and upper bounds to γ_s^* . In contrast, the problem addressed in the paper is the exact computation of the value of the infimum γ_s^* using a non-iterative method. The *a-priori* knowledge of γ_s^* enables designers to assess how close their H^∞-norm bound designed are to the actual optimum performance. In the next section we shall recall the definition of the special coordinate basis (s.c.b) for a linear time-invariant non-strictly proper system [15]. Such a coordinate basis has a distinct feature of explicitly displaying the infinite-zero and finite-zero structures of a given system as well as other system geometric properties. It is instrumental in the derivation of the numerical algorithm.

3. SPECIAL COORDINATE BASIS

In the following we recapitulate the main results in a theorem and some properties of the special coordinate basis while leaving detailed derivation and proofs to be found in [15]. Consider the system described in equations (3.a)-(3.c). It can be easily shown that using singular value decomposition one can always find an orthogonal transformation U and a nonsingular matrix V that put the direct feedthrough matrix D into the following form

$$\overline{D} = UDV = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}, \tag{4}$$

where r is the rank of D. Without loss of generality one can assume that the matrix D in equation (3.c) has the form as shown in equation (4). Thus the system in equations (3.a)-(3.c) can be rewritten as

$$\dot{x} = Ax + [\,B_0 \quad B_1\,] \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} + Ew, \tag{5.a}$$

$$\begin{bmatrix} z_0 \\ z_1 \end{bmatrix} = \begin{bmatrix} C_0 \\ C_1 \end{bmatrix} x + \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix}, \tag{5.b}$$

where B_0, B_1, C_0 and C_1 are the matrices of appropriate dimensions. Note that the inputs u_0 and u_1, and the outputs z_0 and z_1 are those of the transformed system. Namely,

$$u = V \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} = Uz.$$

Note that the H^∞-norm of the system transfer function $T_{zw}(s)$ is unchanged when we apply an orthogonal transformation on the output z, and under any nonsingular transformations on the states and control inputs. We have the following main theorem.

Theorem (s.c.b) [15] *There exist non-singular transformations* Γ_s, Γ_o *and* Γ_i *such that*

$$x = \Gamma_s[x_a', x_b', x_c', x_f']', \quad x_a = [(x_a^-)', (x_a^+)']'$$

$$[z_0', z_1']' = \Gamma_o[z_0', z_f', z_b']',$$

$$[u_0', u_1']' = \Gamma_i[u_0', u_f', u_c']'$$

and

$$\Gamma_s^{-1}(A - B_0C_0)\Gamma_s = \begin{bmatrix} A_{aa}^- & A_o & L_{ab}^-C_b & 0 & L_{af}^-C_f \\ 0 & A_{aa}^+ & L_{ab}^+C_b & 0 & L_{af}^+C_f \\ 0 & 0 & A_{bb} & 0 & L_{bf}C_f \\ B_cE_{ca}^- & B_cE_{ca}^+ & B_cE_{cb} & A_{cc} & L_{cf}C_f \\ B_fE_{fa}^- & B_fE_{fa}^+ & B_fE_{fb} & B_fE_{fc} & A_{ff} \end{bmatrix}, \tag{6.a}$$

$$\Gamma_s^{-1}\,[\,B_0 \quad B_1\,]\,\Gamma_i = \begin{bmatrix} B_{0a}^- & 0 & 0 \\ B_{0a}^+ & 0 & 0 \\ B_{0b} & 0 & 0 \\ B_{0c} & 0 & B_c \\ B_{0f} & B_f & 0 \end{bmatrix}, \tag{6.b}$$

$$\Gamma_o^{-1}\begin{bmatrix} C_0 \\ C_1 \end{bmatrix}\Gamma_s = \begin{bmatrix} C_{0a}^- & C_{0a}^+ & C_{0b} & C_{0c} & C_{0f} \\ 0 & 0 & 0 & 0 & C_f \\ 0 & 0 & C_b & 0 & 0 \end{bmatrix}, \quad \Gamma_o^{-1} D \Gamma_i = \begin{bmatrix} I_r & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{6.c}$$

The proof of this theorem can be found in reference [15]. We note that the output transformation Γ_o can be further partitioned as

$$\Gamma_o = \begin{bmatrix} I_r & 0 \\ 0 & \Gamma_{or} \end{bmatrix} \tag{7}$$

and C_f can be rearranged using the state transformation Γ_s such that $C_f = [0,\ I_{m_f}]$ where m_f is the rank of C_f. In what follows, we state some important properties of the s.c.b which are pertinent to our present work.

Property 1 *The given system (A, B, C, D) is right-invertible if and only if x_b and hence z_b are nonexistent, left-invertible if and only if x_c and hence u_c are nonexistent, invertible if and only if both x_c and x_b are nonexistent.*

Property 2 *Invariant zeros of (A, B, C, D) are the eigenvalues of A_{aa}^- and A_{aa}^+. Moreover, the minimum-phase and nonminimum-phase invariant zeros of (A, B, C, D) are the eigenvalues of A_{aa}^- and A_{aa}^+ respectively.*

Property 3 *The pair (A, B) is stabilizable if and only if (A_{con}, B_{con}) is stabilizable where*

$$A_{con} = \begin{bmatrix} A_{aa}^+ & L_{ab}^+ C_b \\ 0 & A_{bb} \end{bmatrix}, \quad B_{con} = \begin{bmatrix} B_{0a}^+ & L_{af}^+ \\ B_{0b} & L_{bf} \end{bmatrix}. \tag{8}$$

Property 4 *If the system (A, B, C, D) is stabilizable and right-invertible, i.e., x_b is nonexistent, then the pair $(A_{aa}^+,\ [B_{oa}^+,\ L_{af}^+])$ is controllable.*

There are interconnections between the s.c.b and various invariant and almost invariant geometric subspaces. To establish these interconnections, let us define the following subspaces:

- $\mathcal{V}^g(A, B, C, D)$ — the maximal subspace of $\Re^n$ which is $(A + BF)$–invariant and contained in $Ker(C + DF)$ such that the eigenvalues of $(A + BF)|\mathcal{V}^g$ are contained in $\mathcal{C}_g \subseteq \mathcal{C}$ for some F.
- $\mathcal{S}^g(A, B, C, D)$ — the minimal $(A + KC)$–invariant subspace of $\Re^n$ containing $Im(B + KD)$ such that the eigenvalues of the map which is induced by $(A + KC)$ on the factor space $\Re^n / \mathcal{S}^g$ are contained in $\mathcal{C}_g \subseteq \mathcal{C}$ for some K.

For the cases that $\mathcal{C}_g = \mathcal{C}$, $\mathcal{C}_g = \mathcal{C}^-$ and $\mathcal{C}_g = \mathcal{C}^o \cup \mathcal{C}^+$, we replace the index g in $\mathcal{V}^g$ and $\mathcal{S}^g$ by '$*$', '$-$' and '$+$', respectively.

We list in the following the geometrical interpretations of some state vector components of s.c.b.

Property 5

$x_a^- \oplus x_a^+ \oplus x_c$ spans $\mathcal{V}^*(A, B, C, D)$.
$x_a^- \oplus x_c$ spans $\mathcal{V}^-(A, B, C, D)$.
$x_a^+ \oplus x_c$ spans $\mathcal{V}^+(A, B, C, D)$.
$x_c \oplus x_f$ spans $\mathcal{S}^*(A, B, C, D)$.
$x_a^- \oplus x_c \oplus x_f$ spans $\mathcal{S}^+(A, B, C, D)$.
$x_a^+ \oplus x_c \oplus x_f$ spans $\mathcal{S}^-(A, B, C, D)$.

4. COMPUTATIONAL ALGORITHM FOR γ_s^*

In this section we give a simple non-iterative procedure for determining γ_s^*. The method assumes that the system (A, B, C, D) is stabilizable, right-invertible and has no invariant zeros in $\mathcal{C}^o$. The assumption of right-invertibility is, for a nondegenerate case, equivalent to one where the number of control inputs must be greater than or equal to the number of controlled outputs. This is not restrictive since in practice the number of controlled variables is almost always defined to be equal to the number of control elements. The other assumption on the invariant zeros is typical in H^∞literature.

Before we give the proof of our results, let us outline the steps involved in the computation of γ_s^*,

Step 1: Transform the system (A, B, C, D) into the special coordinate basis s.c.b described in section 3 and apply the state transformation matrix Γ_s to the disturbance input distribution matrix E as follows,

$$\Gamma_s^{-1}E = \begin{bmatrix} E_a^- \\ E_a^+ \\ E_c \\ E_f \end{bmatrix} \tag{9}$$

Note that the component associated with x_b is missing since x_b is nonexistent for a right-invertible system.

Step 2: If the system (A, B, C, D) is of nonminimum phase then solve the following Lyapunov equations,

$$A_{aa}^+ S + S(A_{aa}^+)' = [B_{0a}^+,\ L_{af}^+\Gamma_{or}^{-1}]\cdot[B_{0a}^+,\ L_{af}^+\Gamma_{or}^{-1}]', \tag{10.a}$$

$$A_{aa}^+ T + T(A_{aa}^+)' = E_a^+(E_a^+)'. \tag{10.b}$$

The existence and uniqueness of the solutions S and T of the above Lyapunov equations follow from the fact that $\lambda(A_{aa}^+) \in \mathcal{C}^+$ (i.e., $-A_{aa}^+$ is a stable matrix) where the eigenvalues of are the right-half plane invariant zeros of the system (A, B, C, D). Moreover, from the property 4 of section 3, the pair $(A_{aa}^+,\ [B_{oa}^+,\ L_{af}^+\Gamma_{or}^{-1}])$ is controllable when the system (A, B, C, D) is stabilizable and right-invertible. The solution S of equation (10.a) is therefore positive definite and hence invertible.

Step 3: The infimum γ_s^* under state-feedback control is given by

$$\gamma_s^* = \begin{cases} \sqrt{\lambda_{max}(TS^{-1})} & \text{if } (A, B, C, D) \text{ is of nonminimum phase;} \\ 0 & \text{if } (A, B, C, D) \text{ is of minimum phase.} \end{cases} \tag{11}$$

Noting that $\lambda_{max}(TS^{-1}) = 0$ if and only if $T = 0$. This condition occurs when $E_a^+ = 0$ in equation (10.b).

Main Theorem *Consider the system of (3.a)-(3.c) and suppose that (A, B, C, D) is stabilizable, right-invertible and possesses no invariant zeros on $\mathcal{C}^o$. Then γ_s^* as given in equation (11) is the infimum of the H^∞optimal control under state feedback.*

Proof The proof will appear elsewhere.

It is worthwhile to remark that the results obtained by Petersen [9] are a subset of ours. In his formulation, he treated a system where the direct feedthrough matrix D is nonsingular, implying that the system is square invertible and has no infinite zeros.

Corollary *The infimum* γ_s^* *is equal to zero if and only if* $Im(E) \subseteq \mathcal{S}^+(A,B,C,D)$.

Proof If the system (A,B,C,D) is of minimum phase, $\gamma_s^*=0$ and for minimum-phase system $Im(E)$ is always contained in $\mathcal{S}^+(A,B,C,D)$. Conversely, if $Im(E) \not\subseteq \mathcal{S}^+(A,B,C,D)$ which implies $E_a^+ \neq 0$ then it is trivial to see from equation (10.b) that $T \neq 0$ and hence $\gamma_s^* \neq 0$.

In the next section we will provide four numerical examples illustrating the computation of γ_s^*.

5. NUMERICAL EXAMPLES

We will demonstrate the accuracy of the computation of γ_s^* with two examples in aircraft control, one in control of a flexible mechanical system and one illustrating the result of the above corollary.

Example 1: The first example involves the minimization of the H$^\infty$-norm of the aircraft normal acceleration n_{zcg}(g) response to longitudinal turbulence u_g(fps) for a B767 longitudinal aircraft model using elevator control δ_e(deg). State model of this system is of sixth-order containing four rigid-body states $\{V(\text{fps}),\ \alpha(\text{deg/s}),\ q(\text{deg/s}),\ \theta(\text{deg})\}$, one actuator state $x_{\delta e}$ and one Dryden turbulence filter state x_{ug}. It is given by the following state equation for a flight condition of mach 0.80, altitude 35,000ft and center of gravity at 0.18MAC (mean aerodynamic chord):

$$\begin{bmatrix} \dot V \\ \dot\alpha \\ \dot q \\ \dot\theta \\ \dot x_{ug} \\ \dot x_{\delta e} \end{bmatrix} = \begin{bmatrix} -1.675\times10^{-2} & 0.11214 & 2.8\times10^{-4} & -0.56083 & 1.675\times10^{-2} & -2.432\times10^{-2} \\ -1.64\times10^{-2} & -0.77705 & 0.99453 & 1.47\times10^{-3} & 1.64\times10^{-2} & -6.339\times10^{-2} \\ -4.167\times10^{-2} & -3.6595 & -0.95443 & 0 & 4.167\times10^{-2} & -3.6942 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.4447 & 0 \\ 0 & 0 & 0 & 0 & 0 & -15 \end{bmatrix} \begin{bmatrix} V \\ \alpha \\ q \\ \theta \\ x_{ug} \\ x_{\delta e} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 15 \end{bmatrix} \delta_e + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.9431 \\ 0 \end{bmatrix} u_g,$$

$$n_{zcg} = [6.94\times10^{-3}\ \ 0.32795\ \ 2.31\times10^{-3}\ \ 0\ \ -6.94\times10^{-3}\ \ 2.679\times10^{-2}] \cdot \begin{bmatrix} V \\ \alpha \\ q \\ \theta \\ x_{ug} \\ x_{\delta e} \end{bmatrix}.$$

The transfer function from elevator control input δ_e to aircraft normal acceleration n_{zcg} at the center of gravity is invertible and inherently nonminimum-phase. The invariant zeros of this system are: $\{-6.7743,\ 6.13546,\ -8.1557\times10^{-3},\ -4.3926\times10^{-4},\ -0.4447\}$. Note that the system has a nonminimum-phase zero at 6.13546 rad/s. The best achievable reduction in H$^\infty$-norm of the transfer function between the disturbance input u_g and the controlled output n_{zcg} is given by the infimum γ_s^* . We obtain

$$\gamma_s^* = 8.501151331329 \times 10^{-4}.$$

Equations (10.a) and (10.b) reduce simply to two scalar Lyapunov equations since the system possesses only one nonminimum-phase invariant zero. The solutions of these equations are as follows: $S = 2354.183571325698$ and $T = 0.00170135843719$ where $A_{aa}^+ = 6.13546001852626$, B_{0a}^+ does not exist since the system is strictly proper, $L_{af}^+ = -169.9647$, $\Gamma_{or} = 1$ and $E_a^+ = -0.14448956134301$.

Example 2: The second example is the control of a flexible mechanical system consisting of four discs connected by flexible rods. The control input is a torque u applied to the leftmost disc while the controlled variable is the angular position z of the rightmost disc. A torque disturbance w is applied to the third disc. The discs are assumed to have equal rotational inertia J and the connecting rods to have equal torsional stiffness k. Time is measured in units of $\sqrt{J/k}$. The state model of this system is of eighth-order and it is given by

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} w,$$

$$z = [\,1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\,]\, x.$$

The system is invertible and has no invariant zeros; hence it is of minimum-phase. The infimum of H$^\infty$-optimization with state feedback is therefore equal to zero, i.e., $\gamma_s^* = 0$. Again it is difficult to determine this solution using an ARE-based method. Usually the Riccati-solver tends to break down even when γ is not close to γ_s^* .

Example 3: This example is a disturbance rejection design for a fighter aircraft. The problem is to minimize the H$^\infty$-norm of aircraft airspeed and normal acceleration responses $\{V_a(\text{fps}),\ n_{zcg}(\text{g})\}$ to longitudinal and transverse turbulences $\{u_g(\text{fps}),\ w_g(\text{fps})\}$ using both the elevator and throttle controls $\{\delta_e(\text{deg}),\ \delta_{th}\ (\%\ \text{power lever angle})\}$. It is a multi-input multi-output system. The model is for an AFTI-F16 longitudinal aircraft flying at mach 0.90 and altitude of 20,000ft. State model including the Dryden turbulence filters and a first-order actuator model for the throttle control is of eighth-order, and it is given by

$$\dot{x} = \left[\begin{matrix} -0.011669 & 0.024753 & -0.5271 & -0.5601 & 0.001496 & 0.011669 \\ -0.044669 & -1.437 & 16.14 & -0.018286 & -1.065 \times 10^{-7} & 0.044669 \\ -0.088168 & -0.080495 & -0.7046 & 0 & 7.2141 \times 10^{-7} & 0.088168 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.5303 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \right.$$

$$\left.\begin{matrix} -0.024753 & 0 \\ 1.437 & 0 \\ 0.080495 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.5303 & 0.005303 \\ -0.005303 & -0.5303 \end{matrix}\right] x + \begin{bmatrix} 0.017112 & 0 \\ -2.304 & 0 \\ -21.7 & 0 \\ 0 & 0 \\ 0 & 99.06 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_{th} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1.03 & 0 \\ 0 & 1.261 \\ 0 & -53.32 \end{bmatrix} \begin{bmatrix} u_g \\ w_g \end{bmatrix},$$

$$\begin{bmatrix} V_a \\ n_{zcg} \end{bmatrix} = \left[\begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 7.3647 \times 10^{-4} & 0.044067 & -0.003463 & -1.298 \times 10^{-6} & 8.6435 \times 10^{-9} \end{matrix}\right.$$

$$\left.\begin{matrix} -1 & 0 & 0 \\ -7.3647 \times 10^{-4} & -0.044067 & 0 \end{matrix}\right] x + \begin{bmatrix} 0 & 0 \\ -0.044389 & 0 \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_{th} \end{bmatrix}.$$

This two-input-two-output system is invertible, unstable and nonminimum-phase with invariant zeros at $\{-1.3692 \pm 18.637i,\ 1.1338 \times 10^{-3},\ -0.5303,\ -0.5303 \pm 5.3030 \times 10^{-3} i\}$. Our algorithm yields the following value of

$$\gamma_s^* = 4.811041602735 \times 10^{-5}.$$

Again the result is obtained from solutions of two scalar Lyapunov equations since this system has only one nonminimum-phase invariant zero at 1.1338×10^{-3}rad/s. The solutions of these equations are as follows: $S = 2.48122600699 \times 10^{8}$ and $T = 0.57430758138$ where $A_{aa}^{+} = 0.00113375406347$, $B_{0a}^{+} = -750.0799932101727$, $L_{af}^{+} = -0.13170630262325$, $\Gamma_{or} = 1$ and $E_a^{+} = [0.00000165569911, -0.03608666104546]$. The ARE-based method will produce an upper bound on γ_s^* of 4.8472×10^{-5} after numerous trials.

Example 4: Consider a system characterized by

$$\dot{x} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 1 & 1 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} w,$$

$$z = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} u.$$

One can easily check that this system is square, invertible, controllable and observable with two nonminimum-phase invariant zeros at $\{1, 2\}$. Furthermore this system is already in the form of s.c.b and $E_a^{+} = [0 \ \ 0]'$. Hence it follows from the corollary that $\gamma_s^* = 0$.

6. CONCLUSIONS

In this paper we have presented a simple, accurate and non-iterative algorithm for the computation of the infimum in the standard H^∞-optimization problem via state feedback. The results are applicable only to right-invertible systems with no invariant zeros on the imaginary axis. These assumptions are not restrictive as demonstrated in the example section with two common aircraft control problems, and a control problem of undamped flexible mechanical system. The procedure involves solutions of two Lyapunov equations usually of low dimensionality determined from the number of nonminimum-phase finite invariant zeros. The algorithm for the computation of the infimum in H^∞-optimization using output feedback is the subject of [16].

REFERENCES

[1]. Zames, G., "Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms, and Approximate Inverses," IEEE Trans. Automat. Contr., Vol. AC-26, No. 2, pp. 301-320, 1981.

[2]. Doyle, J.C., *Lecture Notes in Advances in Multivariable Control*, ONR/Honeywell Workshop, 1984.

[3]. Francis, B.A., *A Course in H^∞ Control Theory*, Volume 88 of Lecture Notes in Control and Information Sciences, Springer-Verlag, Berlin, Heidelberg, 1987.

[4]. Glover, K., "All Optimal Hankel-Norm Approximations of Linear Multivariable Systems and Their L_∞-Error Bounds," Int. J. Contr., Vol. 39, pp. 1115-1193, 1984.

[5]. Doyle, J.C., Glover, K., Khargoneka, P.P. and Francis, B.A., "State-space Solutions to Standard H_2 and H^∞Control Problems," IEEE Trans. Automat. Contr., Vol. AC-34, No.8, pp.831-847, 1989.

[6]. Doyle, J.C., Glover, K., "State-Space Formulae for All Stabilizing Controllers that Satisfy an H^∞-Norm Bound and Relations to Risk Sensitivity," Systems and Control Letters, 11, 1988, pp. 167-172, North Holland.

[7]. Khargonekar, P., Petersen, I.R. and Rotea, M.A., "H^∞-Optimal Control with State Feedback," IEEE Trans. Automat. Contr., Vol. AC-33, pp. 786-788, 1988.

[8]. Petersen, I.R., "Disturbance Attenuation and H^∞-Optimization: A Design Method Based on the Algebraic Riccati Equation," IEEE Trans. Automat. Contr., Vol. AC-32, No. 5, pp. 427-429, 1987.

[9]. Petersen, I.R., "Complete Results for a Class of State Feedback Disturbance Attenuation Problems," Proceedings of the 27th CDC, Austin, Texas, pp. 1349-1353, 1988.

[10]. Stoorvogel, A.A., "The Singular H^∞Control Problem with Dynamic Measurement Feedback," Preprint 1989.

[11]. Stoorvogel, A.A. and Trentelman, H.L., "The Quadratic Matrix Inequality in Singular H^∞Control with State Feedback," SIAM J. Control and Optimization, Vol. 28, No. 5, pp.1190-1208, September 1990.

[12]. Tadmor, G., "H^∞in the Time Domain: The Standard Four Blocks Problem," Preprint 1988.

[13]. Zhou, K. and Khargonekar, P., "An Algebraic Riccati Equation Approach to H^∞-Optimization," Systems and Control Letters 11, pp. 85-91, 1988.

[14]. Papavassilopoulos, G.P. and Safonov, M.G., "Robust Control Design via Game Theoretic Methods," Proceedings of the CDC, Tampa, Florida, pp.382-387, 1989.

[15]. Saberi, A. and Sannuti, P., "Squaring Down of Non-Strictly Proper Systems," Int. J. Contr., 51, 3, pp.621-629, 1990.

[16]. Chen, B.M., Saberi, A. and Ly, U., "Computation of the Infimum in H^∞-Optimization via Output Feedback," Washington State University Technical Report ECE-111, September, 1990.

ROBUST DESIGN FOR ILL-CONDITIONED PLANTS WITH DIAGONAL INPUT UNCERTAINTY

JIE CHEN AND JAMES S. FREUDENBERG
The Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, MI 48109

ABSTRACT

We consider the design problem of maintaining the output sensitivity function small despite the presence of two blocks of uncertainty at the plant input. Our objective is to develop loop-shaping design strategies to achieve robust performance. We first show in general that plant ill-conditioning imposes special requirements upon the size and shape of the open loop transfer function matrix, and that these requirements are dictated by both the condition number and the input directionality properties of the plant. Secondly, we study a special class of compensators that are constructed according to these requirements. The special structure of such a compensator leads to design guidelines similar to rules of thumb used in classical loop-shaping method. A design is performed for an example to illustrate our results.

1. INTRODUCTION

A problem of significant practical interest in feedback design is to insure stability and performance robust against multiple sources of modelling uncertainty. A useful stability margin for such problems is furnished by the structured singular value (SSV) [8], [10]. For the purpose of design, it is further possible to combine SSV analysis with H_∞ optimization [11], [7] to yield a so-called 'μ-synthesis' procedure [9], [16]. However, because the SSV is a complicated function of the plant, compensator and uncertainty description and can only be computed via a numerical optimization procedure, it is sometimes difficult to obtain insights important for design. Furthermore, it has been observed that the μ-synthesis may sometimes fail to produce a satisfactory solution, particularly when the plant transfer function matrix is *ill-conditioned* [16].

An alternative approach to address these design problems has been seeked to utilize bounds upon the SSV [12], [13], [5], [6]. In [12], [13], the problem of maintaining the output sensitivity function small in the presence of an unstructured uncertainty at the plant input has been studied. Analysis of this problem in these references and in [17], [18], [19] exposed potential design difficulties associated with ill-conditioned plants. The analysis of [12], [13] in particular revealed useful insights upon how these difficulties may arise and how they may be coped with in design. Continuing this work, the authors studied in [5] the problem of maintaining the output sensitivity function small despite the presence of two blocks of uncertainty at the plant input. Potential design difficulties for this problem were shown to be dependent on the plant condition number and its input directionality properties.

The purpose of this paper is to continue the investigation of [5], with a particular emphasis on developing loop-shaping strategies useful in resolving the design difficulties. In Section 2, we shall briefly describe our problem and outline our approach. In Section 3, we derive necessary conditions that are to be satisfied by the open loop transfer function in order to meet the robust performance objective. Our design strategy is to shape the open loop transfer function so that these conditions are satisfied sufficiently well. Section 4 considers a special class of compensators that are constructed according to this strategy, whose structure makes loop-shaping task particularly simple. Specifically, it allows us to obtain useful new bounds upon the SSV, and analysis of these bounds leads to rules of thumb for design. Finally, in Section 5, we illustrate our results by designing a compensator for an example. Section 6 concludes our discussion.

2. PRELIMINARIES

We now briefly describe our problem. Given a nominally stable linear time-invariant feedback system, we denote by $P(s)$ and $F(s)$ the transfer functions of the plant model and compensator, respectively. We shall assume[1] that the plant is square, and that $\det P(s) \not\equiv 0$. Define the *input open loop transfer function, sensitivity function, and complementary sensitivity function* by $L_I(s) = F(s)P(s)$, $S_I(s) = [I + L_I(s)]^{-1}$, $T_I(s) = L_I(s)[I + L_I(s)]^{-1}$, and the *output open loop transfer function, sensitivity function, and complementary sensitivity function*: $L_O(s) = P(s)F(s)$, $S_O(s) = [I + L_O(s)]^{-1}$, $T_O(s) = L_O(s)[I + L_O(s)]^{-1}$. Suppose that the true plant is given by

$$\tilde{P}(s) = [I + r_3(s)\Delta_3(s)]^{-1}P(s)[I + R_I(s)\Delta_I(s)]. \tag{2.1}$$

Our goal is to design a compensator so that the system is robustly stable in the presence of uncertainties $\Delta_I(s)$ and $\Delta_3(s)$. Here $\Delta_I(s) = \mathrm{diag}[\Delta_1(s), \Delta_2(s)]$, $\Delta_1(s) \in \mathbf{C}^{\mathbf{k}\times\mathbf{k}}$, $\Delta_2(s) \in \mathbf{C}^{(\mathbf{n}-\mathbf{k})\times(\mathbf{n}-\mathbf{k})}$, $\Delta_3(s) \in \mathbf{C}^{\mathbf{n}\times\mathbf{n}}$, and $0 \le k \le n$. Define $\Delta = \mathrm{diag}[\Delta_1, \Delta_2, \Delta_3]$, and assume that Δ lies in the set

$$\mathbf{\Delta}_\gamma = \{\, \Delta = \mathrm{diag}[\Delta_1, \Delta_2, \Delta_3] : \ \Delta \text{ stable}, \ \|\Delta\|_\infty < \gamma \,\}, \tag{2.2}$$

where $\|\Delta\|_\infty = \sup_\omega \bar{\sigma}[\Delta(j\omega)]$. The matrix $R_I(s) = \mathrm{diag}[r_1(s)I_k, \ r_2(s)I_{n-k}]$ is dimensioned compatibly with $\Delta_I(s)$ (I_k and I_{n-k} are identity matrices). The weighting functions $r_1(s)$, $r_2(s)$ and $r_3(s)$ are chosen to be stable and to have stable inverses. Note in particular that in (2.1), Δ_3 is often introduced as a *fictitious* uncertainty to model the disturbance attenuation ability provided by the system, and the robust stability problem against Δ_I and Δ_3 simultaneously has often been referred to as robust disturbance attenuation problem (cf. [1], [5], [10], [17], [18], [19]). A useful tool for this problem is provided by SSV analysis. To apply this technique, we first define an *interconnection matrix* M with respect to $\Delta \in \mathbf{\Delta}_\gamma$ [5]:

$$M = \begin{bmatrix} r_1 E_1^H T_I E_1 & r_1 E_1^H T_I E_2 & -r_1 E_1^H P^{-1} T_O \\ r_2 E_2^H T_I E_1 & r_2 E_2^H T_I E_2 & -r_2 E_2^H P^{-1} T_O \\ -r_3 S_O P E_1 & -r_3 S_O P E_2 & r_3 S_O \end{bmatrix}, \tag{2.3}$$

where $E_1 = \begin{bmatrix} I_k \\ 0 \end{bmatrix}$, $E_2 = \begin{bmatrix} 0 \\ I_{n-k} \end{bmatrix}$. The structured singular value [8], denoted $\mu[M(j\omega)]$, is defined to be the reciprocal of a frequency dependent stability margin with respect to all three sources of uncertainty. Denote $\mu_\infty[M] = \sup_\omega \mu[M(j\omega)]$. Then the system will be robustly stable against any $\Delta \in \mathbf{\Delta}_\gamma$ if and only if

$$\mu_\infty[M] < 1/\gamma. \tag{2.4}$$

Hence, our task is equivalent to requiring (2.4) to be satisfied.

Our approach requires utilizing bounds upon the SSV. To proceed, we first define the following *interaction parameters*. Interpretations of these parameters are found in [5], [6]. Let

$$\mu_1 = \max\left\{\bar{\sigma}[r_1 E_1^H T_I E_1], \ \bar{\sigma}[r_2 E_2^H T_I E_2], \ \bar{\sigma}[r_3 S_O]\right\}. \tag{2.5}$$

It follows from [5] that the system will be robustly stable against each *individual* uncertainty if and only if $\mu_1(j\omega) < 1/\gamma$, $\forall \omega$. Next, define the parameters

$$\mu(1,2) \ = \ \sqrt{\bar{\sigma}[r_1 E_1^H T_I E_2]\bar{\sigma}[r_2 E_2^H T_I E_1]}, \tag{2.6}$$

$$\mu(1,3) \ = \ \sqrt{\bar{\sigma}[r_3 S_O P E_1]\bar{\sigma}[r_1 E_1^H P^{-1} T_O]}, \tag{2.7}$$

$$\mu(2,3) \ = \ \sqrt{\bar{\sigma}[r_3 S_O P E_2]\bar{\sigma}[r_2 E_2^H P^{-1} T_O]}, \tag{2.8}$$

$$\mu_2 = \max_{i,j} \mu(i,j), \qquad i \neq j. \tag{2.9}$$

---

[1] Our results extend to nonsquare plants that satisfy the relevant assumption of left or right invertibility.

As discussed in [5], parameters in (2.6-8) measure how two uncertainties may interact to cause robustness problems. Finally, define

$$\mu(1,2,3) = \left(\bar{\sigma}[r_1 E_1^H T_I E_2]\bar{\sigma}[r_3 S_O P E_1]\bar{\sigma}[r_2 E_2^H P^{-1} T_O]\right)^{1/3}, \tag{2.10}$$

$$\mu(1,3,2) = \left(\bar{\sigma}[r_2 E_2^H T_I E_1]\bar{\sigma}[r_3 S_O P E_2]\bar{\sigma}[r_1 E_1^H P^{-1} T_O]\right)^{1/3}, \tag{2.11}$$

$$\mu_3 = \max_{i,j,h} \mu(i,j,h), \qquad i \neq j \neq h. \tag{2.12}$$

The interaction parameters (2.10-11) measure how three uncertainties may interact to cause robustness problems. Using these parameters, it was shown in [4] that

$$\max\ \{\mu_1, \mu_2, \mu_3\} \leq \mu[M] \leq 3.74 \max\ \{\mu_1, \mu_2, \mu_3\}. \tag{2.13}$$

Hence, it follows from (2.13) that the problem of satisfying (2.4) may be approximately treated as one of maintaining the three parameters μ_1, μ_2, and μ_3 sufficiently small. In particular, it was shown in [5] that the parameters $\mu(i,3)$ and $\mu(i,j,h)$ are particularly problematic. Keeping these parameters small should be of primary concern in our design.

3. LOOP-SHAPING DESIGN STRATEGIES

In this section we develop multivariable loop-shaping design strategies for achieving, if possible, small values of the interaction parameters. We shall first use the *singular value decomposition* (SVD) [15] of the plant to describe its directionality properties [14], [5]. To simplify the exposition, assume that the SVD of the plant, denoted by $P = WTZ^H$, may be partitioned as[2]

$$P = W_1 T_1 Z_1^H + W_2 T_2 Z_2^H \tag{3.1}$$

where $T_1 = \text{diag}[\tau_1, \tau_2, \cdots, \tau_k]$, $T_2 = \text{diag}[\tau_{k+1}, \tau_{k+2}, \cdots, \tau_n]$, $W_1, Z_1 \in \mathbf{C}^{\mathbf{n}\times\mathbf{k}}$, and $W_2, Z_2 \in \mathbf{C}^{\mathbf{n}\times(\mathbf{n}-\mathbf{k})}$. The singular values (i.e., diagonal elements of T) of P are ordered so that $\bar{\sigma}[P] = \tau_1 \geq \tau_2 \geq \cdots \geq \tau_n = \underline{\sigma}[P] \geq 0$, and the columns of W_i and Z_i are orthonormal vectors and thus form an orthonormal basis for a subspace of $\mathbf{C}^{\mathbf{n}}$. The condition number of the plant is defined by $\kappa[P] = \bar{\sigma}[P]/\underline{\sigma}[P]$, and a plant will be called ill-conditioned at a frequency ω if $\kappa[P(j\omega)] \gg 1$. Given any rank k matrix $X \in \mathbf{C}^{\mathbf{n}\times\mathbf{k}}$, we shall denote its column space by $\mathbf{X} \subset \mathbf{C}^{\mathbf{n}}$. Furthermore, We assume in (3.1) that $\tau_l \gg \tau_{l+1}$, and refer to the plant as consisting of a higher gain subsystem ($W_1 T_1 Z_1^H$) and a lower gain subsystem ($W_2 T_2 Z_2^H$) [14] with input directions $\mathbf{Z_i}$ and output directions $\mathbf{W_i}$. We shall also occasionally assume that the gains in each subsystem are *uniform*: $\tau_1 = \tau_k$, and $\tau_{k+1} = \tau_n$, and this will be made clear whenever necessary. Finally, we note that the partition (3.1) need hold only at the frequency in question, and need not correspond to physical subsystems.

We next need to describe how well the plant input directions are aligned with the two blocks of uncertainty. It was shown in [5] that the *principal angles* [2], [15] $\bar{\alpha} = \alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k = \underline{\alpha}$, $\alpha_i \in [0, \pi/2]$, between the subspaces $\mathbf{Z_1}$ and $\mathbf{E_1}$ are a measure of the angular distance between these two subspaces, and thus also between their orthogonal complements $\mathbf{Z_2}$ and $\mathbf{E_2}$. A complete characterization of these angles may be obtained using the *C-S decomposition* and is found in [15]. See also [5] for a generalized version of this decomposition.

We shall now focus on strategies useful for keeping the parameters $\mu(i,3)$ and $\mu(i,j,h)$ small; later developments will show that our strategies for keeping $\mu(i,3)$ small are also useful for maintaining $\mu(i,j,h)$ small. For this purpose, it proves convenient to study the loop-shaping problem defined at the plant input.

Proposition 3.1: Suppose that the uniform gain assumption holds for the plant. Then,
(i) for $i = 1,2$

$$\mu^2(i,3) \geq |r_i r_3| \bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_i^H T_I Z_2] \kappa[P] \tag{3.2}$$

---

[2] Our results in this paper extend to more general partitions. For a discussion of more general cases, see [5].

$$\mu^2(i,3) \leq |r_i r_3| \bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_i^H T_I Z_2] \kappa[P] + \text{ other terms.} \tag{3.3}$$

(ii) Furthermore, for $i \neq j \neq h$,

$$\mu^3(i,j,h) \geq |r_1 r_2 r_3| \bar{\sigma}[E_i^H T_I E_j] \bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_j^H T_I Z_2] \kappa[P] \tag{3.4}$$

$$\mu^3(i,j,h) \leq |r_1 r_2 r_3| \bar{\sigma}[E_i^H T_I E_j] \bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_j^H T_I Z_2] \kappa[P] + \text{ other terms.} \tag{3.5}$$

■

It follows from (3.2) that in order to keep the parameters $\mu(i,3)$ small, it is necessary to keep the products $\bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_i^H T_I Z_2]$, $i = 1,2$, small. Similar statements apply to $\mu(i,j,h)$ and the products $\bar{\sigma}[E_i^H T_I E_j] \bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_j^H T_I Z_2]$. Since the "other terms"[3] in (3.3), (3.5) are *not* proportional to the plant condition number, they are of less significance.

Proposition 3.2: Assume that L_I is nonsingular. Then,

$$\max_{i,j} \bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_j^H T_I Z_2] \geq \frac{1}{2}(1 + \sigma_{k+1}[L_I]^{-1})^{-1}(1 + \sigma_k[L_I])^{-1}. \tag{3.6}$$

(a) If $\underline{\sigma}[L_I] > 1$, then

$$\max_{i,j} \bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_j^H T_I Z_2] \leq \frac{\underline{\sigma}[L_I]}{(\underline{\sigma}[L_I] - 1)^2}. \tag{3.7}$$

(b) If $\bar{\sigma}[L_I] < 1$, then

$$\max_{i,j} \bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_j^H T_I Z_2] \leq \frac{\bar{\sigma}[L_I]}{(1 - \bar{\sigma}[L_I])^2}. \tag{3.8}$$

■

It follows from the upper bound (3.7-8) that the terms $\bar{\sigma}[Z_1^H S_I E_i] \bar{\sigma}[E_j^H T_I Z_2]$ may be kept small by requiring the open loop gains to be either very large in all directions ($\underline{\sigma}[L_I] \gg 1$) or very small in all directions ($\bar{\sigma}[L_I] \ll 1$). Although the former strategy is consistent with typical design requirement at low frequencies, and the latter at high frequencies, there will necessarily be an intermediate frequency range over which neither is appropriate. The lower bound in (3.6) shows that over this frequency range it is *necessary* for the loop transfer function to consist of higher and lower gain subsystems of dimensions compatible with those of the plant. This may be further illustrated using a compensator that explicitly inverts the plant model. Consider a *plant-inverting compensator* [13], [5] of the form

$$F(s) = l(s)P^{-1}(s) \tag{3.9}$$

where $l(s)$ is a scalar transfer function selected so that $F(s)$ is proper. With $F(s)$ given by (3.9), the input and output open loop transfer functions are identical; so are the sensitivity and complementary sensitivity functions. We shall denote these by $L_O(s) = L_I(s) = l(s)I$, $S_O(s) = S_I(s) = s(s)I$, and $T_O(s) = T_I(s) = t(s)I$, where $s(s) = 1/(1 + l(s))$ and $t(s) = l(s)/(1 + l(s))$. It was shown in [5] that if the uniform gain assumption holds for the plant, then for $i = 1,2$, $\mu^2(i,3) \geq |r_i t| \cdot |r_3 s| \kappa[P] \cos \underline{\alpha} \sin \bar{\alpha}$. The following result gives an alternative bound for the SSV with this compensator.

Proposition 3.3: Suppose that the uniform gain assumption holds for the plant. Let F be given by (3.9) and assume that $2k = n$. Then,

$$\begin{aligned} \mu^2[M] \quad \geq \quad & \frac{k}{n}|r_1 t|^2 + \frac{k}{n}|r_2 t|^2 + |r_3 s|^2 + \frac{2}{n}(|r_1 t||r_3 s| + |r_2 t||r_3 s|) \\ & \cdot(\kappa[P] + 1/\kappa[P]) \sum_{i=1}^{k} \cos \alpha_{k-i+1} \sin \alpha_i. \end{aligned} \tag{3.10}$$

---

[3] It should be pointed out that these terms in (3.3) and (3.5) are different, and it is straightforward to calculate them explicitly. Since the resulting expressions are lengthy, they are omitted for the purpose of brevity.

■

Note that in this lower bound, $\alpha_1 = \bar{\alpha}$, and $\alpha_k = \underline{\alpha}$. For later reference, we notice that if $n = 2$, $k = 1$, then $\bar{\alpha}$ and $\underline{\alpha}$ are identical, and we denote this common value by α. The following corollary, reminiscent of a result quoted by Stein [19], is immediate from Proposition 3.3.

Corollary 3.4: Suppose that $n = 2$, $k = 1$, and that $|r_1| = |r_2|$. Let F be given by (3.9). Then

$$\mu^2[M] \geq |r_1 t|^2 + |r_3 s|^2 + 2|r_1 t||r_3 s|(\kappa[P] + 1/\kappa[P]) \cos\alpha \sin\alpha. \tag{3.11}$$

■

It is particularly interesting to examine this lower bound for the case $\alpha = 45^0$. Such an example was analyzed in [17], [5], and will be studied in more detail in Section 5. It follows that

$$\mu^2[M] \geq |r_1 t|^2 + |r_3 s|^2 + |r_1 t||r_3 s|(\kappa[P] + 1/\kappa[P]). \tag{3.12}$$

A comparison of this result with that quoted by Stein in [19] shows that the lower bound in (3.12) coincides with $\mu[M]$ obtained in [19] where it is calculated for a plant-inverting compensator and an unstructured Δ_I. Hence if $\alpha = 45^0$, then for $n = 2$, $k = 1$, and $|r_1| = |r_2|$, the values of $\mu[M]$ will be identical with Δ_I being either unstructured or diagonal. The authors of [17] (pp. 1100) observed this phenomenon for a particular example. It should also be pointed out that the result quoted by Stein holds for arbitrary n.

To summarize, it follows that the value of $\mu_\infty[M]$ cannot be kept small with a plant-inverting compensator unless the quantity

$$\rho[P] = \max_i |r_i r_3| \kappa[P] \cos\underline{\alpha} \sin\bar{\alpha} \tag{3.13}$$

is small near the crossover frequency of the loop transfer function $l(j\omega)$. This important parameter, termed the *robustness indicator*, was first introduced in [5], and indicates potential difficulties in achieving robust performance. Indeed, if the compensator is given by (3.9), then a simple exercise shows that at the crossover frequency ($|l| = 1$), $\mu^2(i,3) \geq \frac{1}{4}|r_i r_3|\kappa[P] \cos\underline{\alpha} \sin\bar{\alpha}$. It is important to note that unlike the problem considered in [13], [14], here a plant-inverting compensator will necessarily result in a large value of $\mu_\infty[M]$ *only* when $\rho[P]$ is large near the crossover, regardless of the size of the plant condition number. The significance of this remark is that $\rho[P]$ depends upon both the plant condition number *and* its directionality properties.

We have thus far demonstrated that it is necessary for the open loop transfer function to be decomposed into high and low gain subsystems dimensioned compatibly with those of the plant. In the remainder of this section, we shall consider a method of manipulating the terms $\bar{\sigma}[Z_1^H S_I E_i]\bar{\sigma}[E_j^H T_I Z_2]$ (and hence manipulating the parameters $\mu(i,3)$ and $\mu(i,j,h)$) by appropriately selecting the singular subspaces of the input loop transfer function. Let

$$L_I = V_1 \Sigma_1 U_1^H + V_2 \Sigma_2 U_2^H \tag{3.14}$$

where $V_1, U_1 \in \mathbf{C}^{\mathbf{n}\times\mathbf{k}}$, $V_2, U_2 \in \mathbf{C}^{\mathbf{n}\times(\mathbf{n}-\mathbf{k})}$, $\Sigma_1 = \mathrm{diag}[\sigma_1, \cdots, \sigma_k]$, and $\Sigma_2 = \mathrm{diag}[\sigma_{k+1}, \cdots, \sigma_n]$. Our first step in shaping the singular subspaces $\mathbf{V_i}$ and $\mathbf{U_i}$ will be to use approximations to the closed loop transfer functions obtained in [14]. Denote these approximations by $S_I = S_I^{(0)} + S_I^{(1)} + O(\epsilon^2)$ and $T_I = T_I^{(0)} + T_I^{(1)} + O(\epsilon^2)$, and consider the open loop transfer function at five different frequency bands which correspond to: (i) large gain in all directions ($\sigma_n \gg 1$, $\epsilon = 1/\sigma_n$); (ii) large gain in only some directions ($\sigma_k \gg 1$, $\epsilon = 1/\sigma_k$); (iii) both large and small gains in different directions ($\sigma_k \gg 1 \gg \sigma_{k+1}$, $\epsilon = \sigma_{k+1} + 1/\sigma_k$); (iv) small gains in only some directions ($\sigma_{k+1} \ll 1$, $\epsilon = \sigma_{k+1}$); (v) small gains in all directions ($\sigma_1 \ll 1$, $\epsilon = \sigma_1$). To illustrate, we shall focus particularly on the case (iii), and analogous analyses hold for other cases. Consider first the problem of shaping $\mathbf{U_i}$. Using the results of [14] (pp. 370), it is straightforward to show that

$$\frac{\bar{\sigma}[V_2^H E_i]\underline{\sigma}[Z_1^H U_2]}{\bar{\sigma}[V_2^H U_2]} \leq \bar{\sigma}[Z_1^H S_I^{(0)} E_i] \leq \frac{\bar{\sigma}[V_2^H E_i]\bar{\sigma}[Z_1^H U_2]}{\underline{\sigma}[V_2^H U_2]},$$

$$\frac{\bar{\sigma}[E_i^H V_1]\underline{\sigma}[U_1^H Z_2]}{\bar{\sigma}[U_1^H V_1]} \le \bar{\sigma}[E_i^H T_I^{(0)} Z_2] \le \frac{\bar{\sigma}[E_i^H V_1]\bar{\sigma}[U_1^H Z_2]}{\underline{\sigma}[U_1^H V_1]}.$$

Hence, an effective way to keep the first order approximations $\bar{\sigma}[Z_1^H S_I^{(0)} E_i]\bar{\sigma}[E_j^H T_I^{(0)} Z_2]$ small is to align $\mathbf{U_i}$ with $\mathbf{Z_i}$: $\mathbf{U_i} = \mathbf{Z_i}$, $i = 1, 2$. Indeed, it follows that $\bar{\sigma}[Z_1^H S_I^{(0)} E_i] = 0$ and $\bar{\sigma}[E_j^H T_I^{(0)} Z_2] = 0$. This allows us to approximate $\bar{\sigma}[Z_1^H S_I E_i]\bar{\sigma}[E_j^H T_I Z_2]$.

Proposition 3.5: Suppose that L_I is partitioned as in (3.14), and let $\mathbf{U_i} = \mathbf{Z_i}$, $i = 1, 2$. Then,

$$\bar{\sigma}[Z_1^H S_I^{(1)} E_i] = \bar{\sigma}[\Sigma_1^{-1}(U_1^H V_1)^{-1} U_1^H E_i], \tag{3.15}$$

$$\bar{\sigma}[E_i^H T_I^{(1)} Z_2] = \bar{\sigma}[E_i^H U_2 (V_2^H U_2)^{-1} \Sigma_2]. \tag{3.16}$$

■

It follows that

$$\frac{\bar{\sigma}[Z_1^H E_i]}{\bar{\sigma}[\Sigma_1]\bar{\sigma}[U_1^H V_1]} \le \bar{\sigma}[Z_1^H S_I^{(1)} E_i] \le \frac{\bar{\sigma}[Z_1^H E_i]}{\underline{\sigma}[\Sigma_1]\underline{\sigma}[U_1^H V_1]},$$

$$\frac{\bar{\sigma}[E_i^H Z_2]\underline{\sigma}[\Sigma_2]}{\bar{\sigma}[V_2^H U_2]} \le \bar{\sigma}[E_i^H T_I^{(1)} Z_2] \le \frac{\bar{\sigma}[E_i^H Z_2]\bar{\sigma}[\Sigma_2]}{\underline{\sigma}[V_2^H U_2]}.$$

Hence for given values of $\bar{\sigma}[\Sigma_1]$ and $\underline{\sigma}[\Sigma_2]$, the smallest values of $\bar{\sigma}[Z_1^H S_I^{(1)} E_i]\bar{\sigma}[E_j^H T_I^{(1)} Z_2]$ can be obtained only by equating the subspaces $\mathbf{V_i}$ and $\mathbf{U_i}$, for $i = 1, 2$. Selecting these subspaces so that

$$\mathbf{U_i} = \mathbf{V_i} = \mathbf{Z_i}, \tag{3.17}$$

it follows readily from (3.15-16) that

$$\frac{\sigma_n}{\sigma_1}\cos\underline{\alpha}\sin\bar{\alpha} \le \bar{\sigma}[Z_1^H S_I^{(1)} E_i]\bar{\sigma}[E_i^H T_I^{(1)} Z_2] \le \frac{\sigma_{k+1}}{\sigma_k}\cos\underline{\alpha}\sin\bar{\alpha}. \tag{3.18}$$

This shows that the products $\bar{\sigma}[Z_1^H S_I^{(1)} E_i]\bar{\sigma}[E_i^H T_I^{(1)} Z_2]$ may be kept small by manipulating the open loop gains. The following result reveals that this strategy is indeed useful for keeping $\bar{\sigma}[Z_1^H S_I E_i]\bar{\sigma}[E_i^H T_I Z_2]$ small.

Proposition 3.6: Suppose in addition to the assumptions of Proposition 3.5, that $\mathbf{V_i} = \mathbf{U_i}$, $i = 1, 2$. Then, for $i = 1, 2$,

$$\bar{\sigma}[Z_1^H S_I E_i]\bar{\sigma}[E_i^H T_I Z_2] \ge \frac{\sigma_n}{(1+\sigma_1)(1+\sigma_n)}\cos\underline{\alpha}\sin\bar{\alpha}. \tag{3.19}$$

Furthermore, if $\sigma_k > 1 > \sigma_{k+1}$, then

$$\bar{\sigma}[Z_1^H S_I E_i]\bar{\sigma}[E_i^H T_I Z_2] \le \frac{\sigma_{k+1}}{(\sigma_k - 1)(1 - \sigma_{k+1})}\cos\underline{\alpha}\sin\bar{\alpha}, \tag{3.20}$$

and for $i \ne j$,

$$\bar{\sigma}[E_i^H T_I E_j]\bar{\sigma}[Z_1^H S_I E_i]\bar{\sigma}[E_j^H T_I Z_2] \le \frac{(\sigma_k - \sigma_{k+1})\sigma_{k+1}}{(\sigma_k - 1)^2(1 - \sigma_{k+1})^2}\cos\underline{\alpha}\sin\bar{\alpha}. \tag{3.21}$$

■

It is now clear from Proposition 3.6 that by aligning the subspaces $\mathbf{V_i}$, $\mathbf{U_i}$ and $\mathbf{Z_i}$, the quantites $\bar{\sigma}[Z_1^H S_I E_i]\bar{\sigma}[E_i^H T_I Z_2]$, and hence the parameters $\mu(i, 3)$ for $i = 1, 2$, can be effectively manipulated through manipulating the gains of the input loop transfer function. In particular, it follows that if the uniform gain assumption is satisfied ($\sigma_1 = \sigma_k \gg 1 \gg \sigma_{k+1} = \sigma_n$), then $\bar{\sigma}[Z_1^H S_I E_i]\bar{\sigma}[E_i^H T_I Z_2] \approx \frac{1}{\kappa[L_I]}\cos\underline{\alpha}\sin\bar{\alpha}$. Furthermore, for $i \ne j$, the inequality

$\bar{\sigma}[E_i^H T_I E_j]\bar{\sigma}[Z_1^H S_I E_i]\bar{\sigma}[E_j^H T_I Z_2] \leq \frac{1}{\kappa[L_I]} \cos \underline{\alpha} \sin \bar{\alpha}$ holds approximately. Hence in this case, a necessary condition for $\mu(i,3) < 1/\gamma$ to hold will be for the loop transfer function to satisfy

$$\kappa[L_I] > \gamma \max_i |r_i r_3| \kappa[P] \cos \underline{\alpha} \sin \bar{\alpha}. \tag{3.22}$$

This indicates that the loop gains in the two subsystems must have a difference proportional to the size of $\rho[P]$, in contrast to the problem in [13], [14] where it was required that

$$\kappa[L_I] > \gamma \max_i |r_i r_3| \kappa[P]. \tag{3.23}$$

Finally, we note from the bound (3.21) and the ensuing discussions that our strategy is also useful for keeping $\mu(i,j,h)$ small. This complements the previous work in [5] where it was shown for the case of $n = 2$, $k = 1$, keeping the parameters μ_2 small will tend to force μ_3 to be small as well.

4. A SPECIAL COMPENSATOR AND $\mu[M]$

In this section we consider a special class of compensators that satisfy the required alignment condition (3.17). Such compensators were originally studied in [13], and they render the task of manipulating the gains more tractable. Our analysis will be limited to studying system properties at a fixed frequency, and we shall assume that the compensator possesses a special structure at that frequency. The problem of obtaining a compensator possessing this structure at *all* frequencies is open.

As in [13], we shall proceed by adopting a non-standard notation for the singular value decomposition of the plant. Specifically, we shall write $P = \tilde{W} R \tilde{Z}^H$, where $R = \text{diag}[p_i]$, $p_i = r_i exp(j\theta_i)$ for some angles θ_i (See [13] for an explanation). It follows that the diagonal elements of R are equal in magnitude to the plant singular values, and are numbered in order of decreasing magnitude. Next, our compensators will be described as

$$F = \tilde{Z} \Gamma \tilde{W}^H \tag{4.1}$$

where $\Gamma = \text{diag}[f_i]$, f_i are allowed to be complex, and are not necessarily numbered in order of decreasing magnitude. The input and output open loop and closed loop transfer functions have the form $L_I = \tilde{Z} \text{diag}[l_i] \tilde{Z}^H$, $S_I = \tilde{Z} \text{diag}[s_i] \tilde{Z}^H$, $T_I = \tilde{Z} \text{diag}[t_i] \tilde{Z}^H$, and $L_O = \tilde{W} \text{diag}[l_i] \tilde{W}^H$, $S_O = \tilde{W} \text{diag}[s_i] \tilde{W}^H$, $T_O = \tilde{W} \text{diag}[t_i] \tilde{W}^H$, where $l_i = p_i f_i$, $s_i = (1 + l_i)^{-1}$, $t_i = l_i(1 + l_i)^{-1}$. The subspace alignment condition (3.17) is clearly satisfied. Furthermore, the particular structure of M with these functions allows us to obtain considerably tighter bounds upon the SSV than those provided by (2.13).

Proposition 4.1: Let F be given by (4.1). Then

$$\mu[M] \geq \max \left\{ \max_{1 \leq i \leq k} |r_1 t_i|, \max_{k+1 \leq i \leq n} |r_2 t_i|, \max_{1 \leq i \leq n} |r_3 s_i| \right\}. \tag{4.2}$$

■

To further simplify our derivation, in the remainder of this section we shall assume that

$$l_1 = \cdots = l_k, \quad l_{k+1} = \cdots = l_n. \tag{4.3}$$

Note that this assumption is somewhat more restrictive than the uniform gain assumption of the loop transfer function.

Proposition 4.2: Suppose in addition to the uniform gain assumption of the plant, that the assumption (4.3) holds. Assume that $2k = n$, and let F be given by (4.1). Then,

$$\begin{aligned} \mu^2[M] \geq\ & \frac{2}{n}(|r_3 s_1||r_1 t_n| + |r_3 s_1||r_2 t_n|)\kappa[P] \sum_{i=1}^{k} \cos \alpha_{k-i+1} \sin \alpha_i \\ & + \frac{2}{n}(|r_3 s_n||r_1 t_1| + |r_3 s_n||r_2 t_1|)(1/\kappa[P]) \sum_{i=1}^{k} \cos \alpha_{k-i+1} \sin \alpha_i \end{aligned} \tag{4.4}$$

$$\begin{aligned}\mu^2[M] \;\le\;& n(\max_i |r_3 s_i| + \max_{i,j} |r_j t_i|)^2 \\ &+ n\sum_{i=1}^{2} \left(|r_i t_n||r_3 s_1|\kappa[P] + |r_i t_1||r_3 s_n|(1/\kappa[P])\right)\cos\underline{\alpha}\sin\bar{\alpha}. \qquad (4.5)\end{aligned}$$

■

Again, for later reference, we state a corollary of Proposition 4.2 for $n = 2$, $k = 1$, and $|r_1| = |r_2|$.

Corollary 4.3: Suppose that $n = 2$, $k = 1$, and that $|r_1| = |r_2|$. Let F be given by (4.1). Then

$$\mu^2[M] \ge 2\left(|r_3 s_1||r_1 t_2|\kappa[P] + |r_3 s_2||r_1 t_1|(1/\kappa[P])\right)\cos\alpha\sin\alpha. \qquad (4.6)$$

$$\begin{aligned}\mu^2[M] \;\le\;& 2(\max_i |r_3 s_i| + \max_i |r_1 t_i|)^2 \\ &+ 4\left(|r_1 t_2||r_3 s_1|\kappa[P] + |r_1 t_1||r_3 s_2|(1/\kappa[P])\right)\cos\alpha\sin\alpha. \qquad (4.7)\end{aligned}$$

■

5. A DESIGN EXAMPLE

The purpose of this section is to illustrate our strategies by designing a compensator for a simple example. The plant we consider is a simplified model of a high-purity distillation column, and was originally studied in [17]. The transfer function of the plant model is given in [17] and may be described by $P = \tilde{W}R\tilde{Z}^H$, where

$$\tilde{W} = \begin{bmatrix} 0.6246 & 0.7809 \\ 0.7809 & -0.6246 \end{bmatrix}, \quad \tilde{Z} = \begin{bmatrix} 0.7066 & 0.7077 \\ -0.7077 & 0.7066 \end{bmatrix}, \qquad (5.1)$$

and $R = \text{diag}[1.972/(75s + 1), 0.01391/(75s + 1)]$. The subspaces do not vary with frequency, and the condition number is constant: $\kappa[P] = 141.7$. The principal angle between the subspaces $\mathbf{z}_i$ and $\mathbf{e}_i$ is given by $\alpha = 45^0$. The true plant is assumed to take the form of (2.1-2), and the uncertainties are scaled so that $\gamma = 1$. The weighting functions used in [17] are $r_1(s) = r_2(s) = 0.2(5s + 1)/(0.5s + 1)$ and $r_3(s) = 0.5(10s + 1)/10s$. Using the Bode plot, it can be seen that the indicator $\rho[P]$ is very large at all frequencies, and thus potential robustness problems are anticipated. Indeed, the design difficulties for this problem were demonstrated in [17] using Bode plots of the SSV, and further discussed in [5] by constructing a plant-inverting compensator.

Our design goals will be to achieve nominal closed loop stability and to satisfy the bound in (2.4), and we shall attempt this by utilizing the design strategies developed in the preceding sections. As pointed out in [13], these strategies are particularly suited to the present example for which the relevant subspaces are constants. An immediate consequence is that it is possible to obtain the compensator structure (4.1) at all frequencies, and by doing so, our design problem reduces to that of designing two separate scalar systems $l_i = p_i f_i$. The latter two problems may each be approached using classical design techniques.

Our guideline in design is to shape the two scalar loops so that the lower bounds upon $\mu[M]$ in (4.2) and (4.6) are kept as small as possible. Specifically, our design requires satisfying

$$|r_3 s_1| < 1, \quad |r_3 s_2| < 1, \qquad (5.2)$$

$$|r_1 t_1| < 1, \quad |r_1 t_2| < 1, \qquad (5.3)$$

$$2|r_3 s_1||r_1 t_2|\kappa[P]\cos\alpha\sin\alpha < 1. \qquad (5.4)$$

Note that here we have weakened the lower bound (4.6) to (5.4) for the purpose of convenience. From Corollary 4.3, this strategy is also effective for keeping the upper bound (4.7) small. It follows from (5.2) that the loop transfer function l_2 should be shaped so that

$$|l_1|,\ |l_2| \gg 1 \longrightarrow |l_2| > |r_3|. \qquad (5.5)$$

This requires that the loop transfer function have high gains at low frequencies so that l_2 crosses over at a frequency higher than ω_2, the crossover frequency for r_3. Likewise, the lower bounds in (5.3) dictate that l_1 to be shaped so that

$$|l_1|,\ |l_2| \ll 1 \longrightarrow |l_1| < |r_1|, \tag{5.6}$$

which requires that the loop transfer function have low gains at high frequencies so that l_1 crosses over at a frequency lower than ω_1, the crossover frequency for r_1. It is important to recognize that the degree of difficulty in attaining (2.4) is inversely proportional to the width $\Delta\omega = \omega_1 - \omega_2$ [19], [13], and thus we shall shape the first loop so that it crosses over near ω_1 and the second loop so that it crosses over near ω_2. Finally, we note that the bound in (5.4) depends upon both l_1 and l_2, and thus the two loops cannot be shaped independently. The additional requirements imposed by (5.4) may be obtained by using approximations of the product $|s_1||t_2|$, obtained in [13]. It follows that in each of five frequency intervals specified in Section 3, (5.4) will tend to be satisfied if the following conditions hold:

$$|l_1|,\ |l_2| \gg 1 \longrightarrow |l_1| > 2|r_1 r_3|\kappa[P]\cos\alpha\sin\alpha, \tag{5.7}$$

$$|l_1| \gg 1,\ |l_2| \approx 1 \longrightarrow |l_1| > 2|r_1 r_3|\kappa[P]\cos\alpha\sin\alpha/|2\sin(\frac{\pi + \angle l_2}{2})|, \tag{5.8}$$

$$|l_1| \gg 1 \gg |l_2| \longrightarrow |l_1|/|l_2| > 2|r_1 r_3|\kappa[P]\cos\alpha\sin\alpha, \tag{5.9}$$

$$|l_1| \approx 1,\ |l_2| \ll 1 \longrightarrow |l_2| < |2\sin(\frac{\pi + \angle l_1}{2})|/(2|r_1 r_3|\kappa[P]\cos\alpha\sin\alpha), \tag{5.10}$$

$$|l_1|,\ |l_2| \ll 1 \longrightarrow |l_2| < 1/(2|r_1 r_3|\kappa[P]\cos\alpha\sin\alpha). \tag{5.11}$$

Hence the loop gains must have a difference large enough to satisfy the requirements due to the plant ill-conditioning.

It is particularly useful to compare the above strategies with those of [13]. We notice that both strategies encompass compensators given in the form of (4.1), and require the loop gains to be ill-conditioned. However, the gap between the loop gains in the present problem is only required to be proportional to the robustness indicator (3.13), in contrast to the problem in [13] where it was required to be proportional to the weighted condition number (see (3.23)). Hence our design effort can be greatly simplified by modifying the compensators obtained in [13], and the modifications may be concentrated on tuning the "spread" between the two loop gains. For this purpose, we adapt the final design of [13] to our problem; the reader is referred to [13] for a complete discussion of this design and how it was obtained. The transfer function of the compensator is given by $F(s) = \tilde{Z}\mathrm{diag}[f_i]\tilde{W}^H$, where

$$f_1 = 24.36\left(\frac{75s+1}{s}\right)\left(\frac{s+0.3}{s+0.03}\right)\left(\frac{s+2}{s+100}\right), \tag{5.12}$$

$$f_2 = 0.9222\left(\frac{75s+1}{s}\right)\left(\frac{s+0.8}{s+0.16}\right) \tag{5.13}$$

The structured singular value has a peak equal to 1.054, which is identical to that of [13] for the case where Δ_I is unstructured. From the above discussion, it seems possible in general that the same peak can be maintained while we reduce the gap between l_1 and l_2. However, it can be shown that even slightly decreasing the gain in the first loop and/or increasing the gain in the second loop will cause a significant increase of $\mu[M]$. Hence the design given by (5.12-13) is also taken as the final design for the present problem.

A possible explanation for the fact that it is difficult to improve upon the design of [13] in the present problem may be the following. In [13], the two scalar loops were shaped essentially for the purpose of satisfying

$$|r_3 s_1||r_1 t_2|\kappa[P] < 1, \tag{5.14}$$

in contrast to the present problem in which we consider satisfying (5.4). However, it is easily observed that (5.4) coincides with (5.14) for $\alpha = 45^0$. It is worth pointing out that the authors of [17] have designed a compensator for this plant using "$\mu-$ synthesis" approach, and they noticed that their compensator yields same values of $\mu[M]$ for Δ_I being either diagonal or unstructured. Also, a simple computation shows that the compensator of [17] possesses the structure of (4.1). Thus, a seemingly reasonable conjecture is that for $n = 2$, $k = 1$, the SSV will be identical regardless of the structure of Δ_I if $\alpha = 45^0$ and the compensator has the form of (4.1). As shown in Section 3, this conjecture is true for plant-inverting compensators (Note that plant-inverting compensators are a subset of those of the form (4.1)). The general case is left for future research.

6. CONCLUSION

We conclude by briefly commenting upon design strategies proposed in this paper. It is noted that our strategies provide only conditions that must be satisfied in design, and the problem of how these strategies may be used to shape the loop transfer function remains to be open. In particular, it should be pointed out that the success of our design in Section 5 depends critically upon the fact that our compensator has the special structure described in Section 4 at *all* frequencies. By using such a compensator, our design problem reduces to that of designing two scalar systems and it has several appealing advantages. First, at a fixed frequency this allows us to manipulate feedback properties using the well-known classical rules of thumb. Next, it is possible to quantify the tradeoff of feedback properties at different frequencies according to the Bode's gain-phase relation [3]. Note, however, that such a luxury is not available to problems where the relevant subspaces vary with frequency. Hence, it remains to determine whether this special type of compensators are desirable in general and if so, how the open loop transfer function may be manipulated at a fixed frequency so that the system has desirable feedback properties at that frequency. Furthermore, it remains necessary to study the tradeoffs that must be performed between feedback properties of multivariable systems at different frequencies.

REFERENCES

[1] J.S. Bird and B.A. Francis, *Proc. 25th IEEE CDC*, pp. 1804-1809, Dec. 1986.

[2] Å. Björck and G. H. Golub, *Mathematics of Computation*, vol. 27, no. 123, pp. 579-594, July 1973.

[3] H.W. Bode, *Network Analysis and Feedback Amplifier Design*, Princeton, NJ: Van Nostrand, 1945.

[4] J. Chen and J. S. Freudenberg, to appear, *Contr.-Theory and Advanced Technology.*

[5] J. Chen and J. S. Freudenberg, *Proc. 28th IEEE CDC*, pp. 441-446, Dec. 1989, also to appear in *IEEE TAC.*

[6] J. Chen and J.S. Freudenberg, *Proc. 1990 ACC*, pp. 435-440, May 1990.

[7] J.C. Doyle, K. Glover, P.P. Khargonekar, and B.A. Francis, *IEEE TAC*, vol. AC-34, no. 8, pp. 831-847, August 1989.

[8] J. C. Doyle, *IEE Proc.*, vol. 129, Pt.D, no. 6, pp. 242-250, Nov. 1982.

[9] J. C. Doyle, *Proc. 24th IEEE CDC*, pp. 260-265, Dec. 1985.

[10] J. C. Doyle, J. E. Wall, and G. Stein, *Proc. 21st IEEE CDC*, pp. 629-636, Dec. 1982.

[11] B.A. Francis, *A course in H_∞ Control Theory*, Springer-Verlag, 1987.

[12] J. S. Freudenberg, *Int. J. Contr.*, vol. 49, no. 3, pp. 851-871, March 1989.

[13] J. S. Freudenberg, *Int. J. Contr.*, vol. 49, no. 3, pp. 873-903, March 1989.

[14] J.S. Freudenberg, *Int. J. Contr.*, vol. 51, no. 2, pp. 365-390, Feb. 1990.

[15] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, 1983.

[16] K.E. Lenz, *Topics Concerning Applied H_∞ Control*, Ph.D. thesis, University of Minnesota, 1988.

[17] S. Skogestad, M. Morari, and J. C. Doyle, *IEEE TAC*, vol. AC-33, no. 12, pp. 1092-1105, Dec. 1988.

[18] G. Stein, Report No. LIDS-P-1504, MIT, Cambridge, MA, August 1985.

[19] G. Stein, *Lecture Notes of Workshop on Robust Multivariable Control*, June 1987.

Robust Stability with Time-varying Perturbations: Andy Packard and Jang Teng, Mechanical Eng. & Electrical Eng. Univ. of California, Berkeley CA, 94720, 415.643.7959

1 Introduction: In this paper, we use the *swapping lemma*, from adaptive control, to give robustness bounds on time-varying, block structured uncertainty.

2 Notation: For $M \in \mathbf{C}^{m \times n}, \bar{\sigma}(M)$ denotes the maximum singular value of M. L_2 deontes the set of measurable, square-integrable functions. For a transfer matrix, $G(s)$ which is analytic in the closed right-half plane, $\|G\|_\infty := \sup_{\omega \in \mathbf{R}} \bar{\sigma}(G(j\omega))$. For general operators from L_2 to L_2, the induced norm will be denoted $\|\cdot\|_{2 \to 2}$. We abuse notation by mixing both time-domain and frequency-domain notions in the same figure, or equation. In figures, addition of time-domain signals is denoted by $\oplus$, while multiplication of time-domain signals is denoted by $\otimes$.

3 Problem Formulation: We consider the familiar interconnection in **Fig. 1** The operator M is assumed to be a known, stable, linear, finite dimensional, time-invariant system, with transfer function $M(s)$. The operator Δ is viewed as a perturbation to the nominal system M. The Δ operator is assumed to belong to a prescribed set, $\mathbf{\Delta}$, and satisfy a norm (or norm-like) bound. A useful uncertainty set consists of block diagonal matrices with real and complex entries, and a prescribed, problem dependent structure of the form,

$$\mathbf{\Delta} := \left\{ \text{diag}\left[\delta_1^R I_{q_1}, \ldots, \delta_S^R I_{q_S}, \delta_1^C I_{p_1}, \ldots, \delta_T^C I_{p_T}, \Delta_1, \ldots \Delta_F\right] : \delta_i^R \in \mathbf{R}, \delta_j^C \in \mathbf{C}, \Delta_k \in \mathbf{C}^{m_k \times m_k} \right\}.$$

In this paper, we derive robust stability bounds for perturbations which are assumed to be **time-varying, complex gains**. The new bounds arise by modifying the existing upper bounds for the **structured singular value** μ,[Doy],[FanT],[FanTD], and give allowable bounds on both the perturbation's magnitude, and its time derivative.

4 Frequency Dependent Scalings and Small Gain Theorem: Consider again, the system in **Fig 1**, where M is a stable, LTI, finite dimensional operator, with transfer function $M(s)$, and Δ is a stable operator, from L_2 to L_2. Let $\beta > 0$. Then from the small gain theorem, the perturbed system is stable for all such Δ with $\|\Delta\|_{2 \to 2} \le \frac{1}{\beta}$ if $\|M(s)\|_\infty < \beta$ In general, this measure for stability robustness is too conservative if Δ represents **structured** as well as bounded uncertainties. Frequency dependent scalings can be employed to reduce this conservativeness. Suppose that $\mathbf{\Delta}$ is simply a unknown matrix of constant complex gains, with prescribed structure $\mathbf{\Delta} := \{\text{diag}\left[\delta_1 I_{q_1}, \delta_2 I_{q_2}, \cdots, \delta_p I_{q_p}\right] : \delta_j \in \mathbf{C}\}$ and $\mathcal{D} := \{\text{diag}\left[D_1(s), D_2(s), \cdots, D_p(s)\right]\}$, *where each* $D_i(s)$ *is a* $q_i \times q_i$ *proper, stable and minimum-phase transfer function.* Since a constant gain commutes with a LTI convolution operator, for any $\Delta \in \mathbf{\Delta}$, $D \in \mathcal{D}$, $D\Delta = \Delta D$, so that the two loops shown in **Fig 1 and 2** are equivalent. Now, using the small gain theorem on the right figure to draw stability conclusions about the left figure gives that the perturbed system is closed-loop stable for all $\Delta \in \mathbf{\Delta}$ with $\bar{\sigma}(\Delta) \le \frac{1}{\beta}$ if there exists a $D(s) \in \mathcal{D}$ such that $\|D(s)^{-1} M(s) D(s)\|_\infty < \beta$ Hence introducing the frequency dependent scalings, $\|D^{-1}(s) M(s) D(s)\|_\infty$ can be significantly lower than $\|M(s)\|_\infty$. Therefore, a less conservative measure for stability robustness is obtained.

5 Difficulties with Time-varying Perturbations: It is well known that time-varying gains do not commute with LTI systems. For example, consider the two systems in **Fig 3 and 4**. Suppose that $u(t)$ is a unit step input, $d(s) = \frac{s+100}{10(s+1)}$, so that $d^{-1}(s) = \frac{10(s+1)}{s+100}$, and that $\delta(t) = \cos 1000t$. Then the steady state outputs of the two systems are $y(t) = \cos 1000t$ and $y_d(t) = 100 \cos 1000t$ respectively. Therefore, the two systems are not equivalent. As a result, **the frequency dependent scalings discussed in the previous section cannot be applied directly to provide a less conservative measure for stability robustness with respect to time-varying structured uncertainties.**

6 Swapping Lemma: A slight modification to the *swapping lemma* from adaptive control [Mor], [SasB], is stated below.

Lemma 6.1: Let $w : \mathbf{R} \to \mathbf{C}^m$ and $\delta : \mathbf{R} \to \mathbf{C}$ and assume that δ is differentiable. Let $\mathcal{L}$, $\mathcal{L}_B$, $\mathcal{L}_C$ denote the causal linear operators from $u \to y$, $u_B \to y_B$, $u_C \to y_C$ respectively with zero initial conditions and state space equations given by

$$\underbrace{\begin{array}{rcl}\dot{x} &=& Ax+Bu\\ y &=& Cx+Du\end{array}}_{\mathcal{L}} \qquad \underbrace{\begin{array}{rcl}\dot{x}_B &=& Ax_B+Bu_B\\ y_B &=& x_B\end{array}}_{\mathcal{L}_B} \qquad \underbrace{\begin{array}{rcl}\dot{x}_C &=& Ax_C+u_C\\ y_C &=& Cx_C\end{array}}_{\mathcal{L}_C}$$

where $A \in \mathbf{C}^{n \times n}$, $B \in \mathbf{C}^{n \times m}$, $C \in \mathbf{C}^{m \times n}$, $D \in \mathbf{C}^{m \times m}$, $u(t), u_B(t), u_C(t), y(t), y_B(t), y_C(t) \in \mathbf{C}^m$, and $x(t) \in \mathbf{C}^n$.

Then, for zero initial conditions, $\mathcal{L}(w\delta) = \mathcal{L}(w)\delta - \mathcal{L}_C(\mathcal{L}_B(w)\dot{\delta})$.

Remark: The proof can be derived, with slight modification, from that found in [Mor],[SasB]. Note that the *swapping lemma* states that the two figures in **Fig 5 and 6** are zero-state equivalent.

7 Application to robustness analysis: Suppose that stable, minimum phase, scaling matrices have been obtained, and a robustness bound for unknown, but constant, complex matrix perturbations has been calculated, as in section 4 . How may the calculation be modified to allow some time-variation in the perturbations? Consider one particular pertubation $\delta(t)$, and the scaling matrix $D(s)$ associated with that perturbation. Let $D(s) := C(sI - A)^{-1}B + E$, $D_B(s) := (sI - A)^{-1}B$, and $D_C(s) := C(sI - A)^{-1}$. Suppose that $D(s)$ is a $q \times q$ rational matrix, and that $A \in \mathbf{C}^{n \times n}$. Employing the *swapping lemma*, and abusing notation, the two systems in **Fig 7 and 8** are equivalent.

Now consider the full problem, with several uncertainties. For each $1 \le i \le p$, assume that $\delta_i : \mathbf{R} \to \mathbf{C}$ are differentiable functions. Given these functions, define matrices $\Delta(t)$ and $\Delta_{tv}(t)$ as

$$\Delta(t) := \operatorname{diag}\left[\delta_1(t)I_{q_1}, \ldots, \delta_p(t)I_{q_p}\right]$$

$$\Delta_{tv}(t) := \operatorname{diag}\left[\delta_1(t)I_{q_1}, \dot{\delta}_1(t)I_{n_1}, \ldots, \delta_p(t)I_{q_p}, \dot{\delta}_p I_{n_p}\right]$$

Finally, let $D_i(s) := E_i + C_i\,(sI - A_i)^{-1}\,B_i$ be stable, minimum phase, $q_i \times q_i$ transfer matrices, with $A_i \in \mathbf{C}^{n_i \times n_i}$. For each i, define $D_{B_i}(s) := (sI - A_i)^{-1}\,B_i$, and $D_{C_i}(s) := C_i\,(sI - A_i)^{-1}$. With these definitions, define the transfer matrix M_D as shown in **Fig 9**. Using the reasoning associated with **Fig 7 and 8**, the two loops in **Fig 10 and 11** are equivalent.

Theorem 7.1: Let matrices $M_D(s), \Delta(t), \Delta_{tv}(t)$ be given as above, and let $\beta > 0$. Suppose that the functions $\delta_i(t)$ satisfy $|\delta_i| \le \frac{1}{\beta}$ and $|\dot{\delta}_i| \le \frac{1}{\beta})$ for all i. If $\|M_D(s)\|_\infty < \beta$ then the system in **Fig 1** is stable.

More generally, let $\alpha_1, \alpha_2, \ldots, \alpha_p$ be positive constants, and define

$$D_{L_\alpha} = \operatorname{diag}\left\{\begin{bmatrix} I_{q_1} \\ \frac{1}{\alpha_1}D_{B_1}\end{bmatrix} D_1^{-1}, \cdots, \begin{bmatrix} I_{q_p} \\ \frac{1}{\alpha_p}D_{B_p}\end{bmatrix} D_p^{-1}\right\}$$

$$D_R = \operatorname{diag}\left\{\left[\begin{array}{cc} D_1 & D_{C_1}\end{array}\right], \cdots, \left[\begin{array}{cc} D_p & D_{C_p}\end{array}\right]\right\}$$

Corollary 7.2: Let $\beta > 0$, and constants α_i be given, with the transfer matrices D_{L_α} and D_R be defined as above. Suppose that the functions $\delta_i(t)$ satisfy $|\delta_i| \le \frac{1}{\beta}$ and $|\dot{\delta}_i| \le \frac{1}{\beta})$ for all i. If $\|D_{L_\alpha} M D_R\|_\infty < \beta$ then the loop in **Fig 1** is stable.

8 Acknowledgements: The authors gratefully acknowledge financial support from the National Science Foundation, award ECS-8909499, State of California MICRO program, and the General Electric Company.

9 References:

Doy: J. Doyle, *IEE Proc.*, vol. 129, Part D, no. 6, pp. 242-250, November, 1982.

DoyWS: J. Doyle, J. Wall, and G. Stein, [*IEEE CDC, pp. 629-636, December, 1982.*

FanT: *M. Fan, and A. Tits, IEEE TAC, vol. AC-31, no. 8, pp. 734-743, 1986.*

FanTD: *M. Fan, A. Tits, and J. Doyle, ACC, pp. 1195-1200, June, 1988.*

Mor: *A.S. Morse, IEEE Trans. on Automatic Control, vol. AC-25, no. 3, pp. 433-439, 1980.*

SasB: *S. Sastry, and M. Bodson, Adaptive Control Prentice-Hall, 1989.*

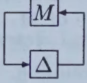


Fig. 1

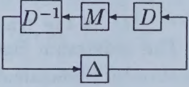


Fig. 2

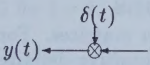

$\delta(t)$
$y(t)$
$u(t)$

Fig. 3

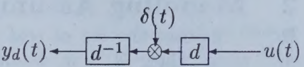

$\delta(t)$
$y_d(t)$
d^{-1}
d
$u(t)$

Fig. 4

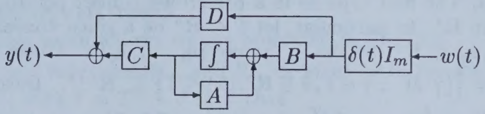

D
$y(t)$
C
∫
B
$\delta(t)I_m$
$w(t)$
A

Fig. 5

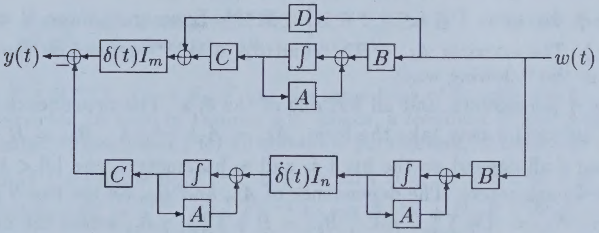

D
$y(t)$
−
$\delta(t)I_m$
C
∫
B
$w(t)$
A
C
∫
$\dot{\delta}(t)I_n$
∫
B
A
A

Fig. 6

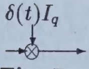


Fig. 7

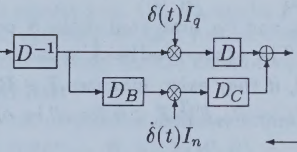

$\delta(t)I_q$
D^{-1}
D
D_B
D_C
$\dot{\delta}(t)I_n$

Fig. 8

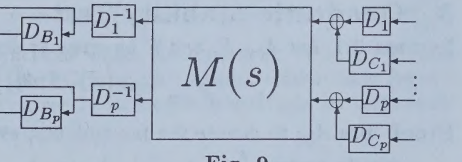

D_{B_1}
D_1^{-1}
$M(s)$
D_1
D_{C_1}
D_{B_p}
D_p^{-1}
D_p
D_{C_p}

Fig. 9

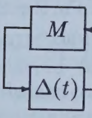


Fig. 10

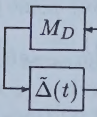

M_D
$\tilde{\Delta}(t)$

Fig. 11

STATE-FEEDBACK SOLUTIONS TO QUADRATIC STABILIZATION PROBLEMS

Andy Packard and Gregory Becker, Dept. of Mech. Engineering, University of California, Berkeley, CA 94720. Support from NSF award ECS-8909499.

1 Abstract

This paper considers a state-feedback, quadratic stabilization problem, where uncertainty in the A and B matrices can depend on both a correlated uncertainty and a special form of linear-fractional uncertainty on real parameters. The existence of a single state-feedback, F, and a single Lyapunov matrix, T, for the entire set of uncertain matrices is reformulated into a convex feasibility problem. This work relies on two previous papers, [BerPG], and [BoyY].

2 Modeling Assumptions

$\mathbf{R}^{m \times n}$ denotes the set of real $m \times n$ matrices. For $M \in \mathbf{R}^{n \times n}, M^T$ is the transpose of M. If $M^T = M$, then $M > 0$, denotes a positive definite matrix. The maximum eigenvalue of M is $\lambda_{\max}(M)$. The column vector, $u = [u_1 \ldots u_n]^T \in \mathbf{R}^n$, has the standard Euclidean norm, $\|u\|$. The uncertainty in the state-space matrices A and B will be modeled with two types of perturbation parameters. The first type lie in a prescribed convex polytope, while the second type lie in the unit cube in $\mathbf{R}^s$. In particular, let $\Gamma \subset \mathbf{R}^p$ be a given convex polytope, with extreme points $\eta_1 \in \mathbf{R}^p, \ldots, \eta_q \in \mathbf{R}^p$, so that $\Gamma = \mathrm{co}(\{\eta_1, \ldots, \eta_q\})$. Given Γ, the set of **allowable parameters** is $\mathcal{P} := \left\{[\gamma \ \ \delta]^T : \gamma \in \Gamma, \delta \in \mathbf{R}^s, |\delta_i| \leq 1\right\} \subset \mathbf{R}^{p+s}$. Denote the 2^s corners of the unit cube in $\mathbf{R}^s$ by $\mathcal{V} := \left\{[\delta_1, \ldots, \delta_s]^T : \delta_i = \pm 1\right\} \subset \mathbf{R}^s$. The **extreme points** of $\mathcal{P}$ is the set, $\mathcal{E} := \left\{[\gamma \ \ \delta]^T : \gamma = \eta_i \text{ for some } 1 \leq i \leq q, \delta \in \mathcal{V}\right\} \subset \mathbf{R}^{p+s}$. Enumerate these $N := q \cdot 2^s$ points as $\mathcal{E} = \{e_1, e_2, \ldots, e_N\}$. The matrices $A_{\delta\gamma} \in \mathbf{R}^{n \times n}$ and $B_{\delta\gamma} \in \mathbf{R}^{n \times m}$ depend on the real parameters $\gamma_1, \ldots \gamma_p, \delta_1, \ldots \delta_s$ in the following way:

- Fix all of the γ parameters, and all but one of the δ_j's. The dependence of $A_{\delta\gamma}$ and $B_{\delta\gamma}$ on the free (unfixed) δ may take the form, $A_{\delta\gamma} = \bar{A} + \frac{\delta}{1-d\delta}\tilde{A}$, $B_{\delta\gamma} = \bar{B} + \frac{\delta}{1-d\delta}\tilde{B}$, where $\bar{A}, \tilde{A}, \bar{B}, \tilde{B}$ and d all depend on the fixed δ_j and γ_i parameters, and $|d| < 1$.
- Fix all of the δ parameters. The dependence of $A_{\delta\gamma}$ and $B_{\delta\gamma}$ on the free γ parameters may take the form, $A_{\delta\gamma} = \hat{A} + \sum_{i=1}^{p} \gamma_i A_i$, $B_{\delta\gamma} = \hat{B} + \sum_{i=1}^{p} \gamma_i B_i$, where the matrices $\hat{A}, \hat{B}, A_i$ and B_i depend on the fixed δ parameters.

Enumerate the values of the matrix functions $A_{\delta\gamma}$ and $B_{\delta\gamma}$ evaluated at the N points of $\mathcal{E}$ as $A_{[i]} := A_{\delta\gamma}|_{e_i}$, and $B_{[i]} := B_{\delta\gamma}|_{e_i}$, for $i = 1, \ldots, N$. The **state-feedback quadratic stabilization problem** is : *Given* Γ, $\mathcal{P}$, $A_{\delta\gamma}$, *and* $B_{\delta\gamma}$. *Find, if they exist, matrices* $T \in \mathbf{R}^{n \times n}, T = T^T > 0$, *and* $F \in \mathbf{R}^{m \times n}$ *such that* $T(A_{\delta\gamma} + B_{\delta\gamma}F) + (A_{\delta\gamma} + B_{\delta\gamma}F)^T T < 0$ *for all* $[\gamma \ \ \delta]^T \in \mathcal{P}$.

3 Quadratic Stability Test

Lemma 3.1 *Let* $A_{\delta\gamma}$, Γ, *and* $\mathcal{P}$ *be given as described in Section 2. Then*

$$\max_{[\gamma_o \ \ \delta_o]^T \in \mathcal{P}} \lambda_{\max}\left(A_{\delta\gamma} + A_{\delta\gamma}^T\right) = \max_{i=1,\ldots,N} \lambda_{\max}\left(A_{[i]} + A_{[i]}^T\right). \tag{3.1}$$

Proof: Use $A_{[o]}$ to denote the function $A_{\delta\gamma}$ evaluated at any point $[\gamma_o \ \ \delta_o]^T \in \mathcal{P}$. For the symmetric matrix $\left(A_{[o]} + A_{[o]}^T\right)$, $\lambda_o := \lambda_{\max}\left(A_{[o]} + A_{[o]}^T\right) = \max_{u \in \mathbf{R}^n, \|u\|=1} u^T\left(A_{[o]} + A_{[o]}^T\right)u = u_o^T\left(A_{[o]} + A_{[o]}^T\right)u_o$ for some $u_o \in \mathbf{R}^n, \|u_o\| = 1$. Fix $\gamma = \gamma_o$ and all but one of the δ_j's that make up δ_o, and denote the the dependence of $A_{\delta\gamma}$ on this δ_j by $A(\delta_j)$. To show that $\max_{\delta_j = \pm 1} \lambda_{\max}\left(A(\delta_j) + A(\delta_j)^T\right) \geq \lambda_o$, define the scalar valued function $f : [-1, 1] \rightarrow \mathbf{R}$ by $f(\delta_j) := u_o^T\left(A(\delta_j) + A(\delta_j)^T\right)u_o$, and notice it does not change sign for all $|\delta_j| \leq 1$. Hence, the maximum value of f must occur at $\delta_j = \pm 1$, and so $\max_{\delta_j = \pm 1} f(\delta_j) \geq \lambda_o$. Also, $\lambda_{\max}\left(A(\delta_j) + A(\delta_j)^T\right) = \max_{u \in \mathbf{R}^n, \|u\|=1} u^T\left(A(\delta_j) + A(\delta_j)^T\right)u \geq u_o^T\left(A(\delta_j) + A(\delta_j)^T\right)u_o = f(\delta_j)$. Thus, $\max_{\delta_j = \pm 1} \lambda_{\max}\left(A(\delta_j) + A(\delta_j)^T\right) \geq \lambda_o$. Repeating for each δ_j, shows for any fixed $\gamma \in \Gamma$, the maximum of $\lambda_{\max}\left(A_{\delta\gamma} + A_{\delta\gamma}^T\right)$ as δ varies over the unit cube in $\mathbf{R}^s$ is achieved when δ is at one of the 2^s points in $\mathcal{V}$. Fix all the δ parameters, and let $\gamma \in \Gamma$ vary. Let $A(\gamma)$ denote the affine dependence of the function $A_{\delta\gamma}$ on γ. Since the function $\mathcal{L} : \mathbf{R}^{n \times n} \to \mathbf{R}$ defined by $\mathcal{L}(A) = \lambda_{\max}\left(A + A^T\right)$ is convex in A, implies the function $\lambda_{\max}\left(A(\gamma) + A(\gamma)^T\right)$ is convex on the convex polytope Γ. By properties of convex functions, the maximum value of the function will occur at one of the extreme points of Γ. Hence, for any point $[\gamma \ \delta]^T \in \mathcal{P}$, there is an extreme point of $\mathcal{P}$, say e_i, such that $\lambda_{\max}\left(A_{[i]} + A_{[i]}^T\right) \geq \lambda_{\max}\left(A_{\delta\gamma} + A_{\delta\gamma}^T\right)$. ♯

Lemma 3.2 *Given* $A_{\delta\gamma}$, *and* Γ, *and* $\mathcal{P}$ *as described in Section* 2, *there is a matrix* $T \in \mathbf{R}^{n \times n}$, *with* $T = T^T > 0$ *such that* $TA_{\delta\gamma} + A_{\delta\gamma}^T T < 0$ *for all* $[\gamma \ \delta]^T \in \mathcal{P}$, *if and only if there is a* $T \in \mathbf{R}^{n \times n}$, *with* $T = T^T > 0$ *such that* $TA_{[i]} + A_{[i]}^T T < 0$ *for* $i = 1, \ldots, N$.

Proof: Let $T \in \mathbf{R}^{n \times n}$, with $T = T^T > 0$ be given. Define a new function $\tilde{A}_{\delta\gamma} := TA_{\delta\gamma}$, and notice that $\tilde{A}_{\delta\gamma}$ has the parameter dependence needed to appeal to Lemma 3.1. ♯

Remark: This lemma is a small extension of the quadratic stability theorem given in [BoyY].

4 State-Feedback Synthesis

The state-feedback synthesis proof follows results presented in [BerPG].

Theorem 4.1 *Let* Γ, $\mathcal{P}$, *and functions* $A_{\delta\gamma}$, *and* $B_{\delta\gamma}$ *be given as in Section 2. There is a matrix* $F \in \mathbf{R}^{m \times n}$ *and* $T \in \mathbf{R}^{n \times n}$, *with* $T = T^T > 0$ *such that*

$$T\left(A_{\delta\gamma} + B_{\delta\gamma}F\right) + \left(A_{\delta\gamma} + B_{\delta\gamma}F\right)^T T < 0 \tag{4.1}$$

for all $[\gamma \ \delta]^T \in \mathcal{P}$ *if and only if there exist* $W \in \mathbf{R}^{n \times n}, W = W^T > 0$, *and* $R \in \mathbf{R}^{m \times n}$ *such that* $A_{[i]}W + WA_{[i]}^T + R^T B_{[i]}^T + B_{[i]}R < 0$ *for all* $i = 1, \ldots, N$.

Proof: For any $F \in \mathbf{R}^{m \times n}$, $A_{\delta\gamma} + B_{\delta\gamma}F$ has the same type of dependence on the parameters as $A_{\delta\gamma}$ was restricted to have in Lemma 3.2. Hence, a feedback F and a symmetric, positive definite T satisfy Equation 4.1 for all allowable parameters, iff the matrices F and T satisfy $T\left(A_{[i]} + B_{[i]}F\right) + \left(A_{[i]} + B_{[i]}F\right)^T T < 0$ for all $i = 1, \ldots, N$. Pre- and post- multiply by T^{-1}, and redefine $W := T^{-1}, R := FT^{-1}$ to get $A_{[i]}W + B_{[i]}R + WA_{[i]}^T + R^T B_{[i]}^T < 0$. ♯

Remark: If such a matrix pair, (W, R), exists, $F = RW^{-1}$ can be used to quadratically stabilize the uncertain linear system $\dot{x} = A_{\delta\gamma}x + B_{\delta\gamma}u$ using $u = Fx$. In this case, one Lyapunov function which establishes the stability for the closed-loop is $\mathbf{V}(x) := x^T W^{-1} x$.

5 Computation

Let $\mathbf{W}$ denote the set of positive definite, symmetric, $n \times n$ matrices. Define the scalar valued function $h : \mathbf{W} \times \mathbf{R}^{m \times n} \to \mathbf{R}$ by $h(W, R) = \max_{i=1,\ldots,N} \lambda_{\max}\left(A_{[i]}W + B_{[i]}R + WA_{[i]}^T + R^T B_{[i]}^T\right)$. Hence, h is a maximum of a finite collection of functions which are convex on $\mathbf{W} \times \mathbf{R}^{m \times n}$, and is convex itself. This is the problem addressed in [BerPG], and we refer the reader there for one particular cutting-plane implementation. We are currently looking into methods for large-scale optimization of eigenvalues of symmetric matrices, [Ove].

6 Conclusions

We considered some simple extensions to the state-feedback quadratic stabilization problem, when the matrices A and B depend on real parameters in a special linear fractional manner. The uncertainty in the A and B matrices may be correlated. The existence of a single state-feedback, and a single quadratic Lyapunov function establishes the stability of the entire set of perturbed closed loop system, and is reformulated as a convex feasibility problem.

7 Reference

[BerPG] Bernussou, Peres, Geromel, "An LP oriented procedure for quadratic stabilization of uncertain systems," *Systems and Control Letters*, vol. 13, pages 65-72, 1989.

[BoyY] S. Boyd, and Q. Yang, "Structured and simultaneous Lyapunov functions for system stability problems," *Int. J. Control*, 1989, vol. 49, No. 6, 2215-2240.

[Ove:] M. Overton, "Large-Scale Optimization of Eigenvalues", NYU Computer Science Department Report, #505, May, 1990.

Complete Instability of a Box of polynomials

Prasad Padmanabhan and C. V. Hollot
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003

Abstract

In this paper we study the converse of Kharitonov's polynomial problem by asking whether the complete instability of a box of polynomials can be determined from extreme sets. We show that it is not enough to check the $(n-4)$-dimensional boundary and prove that the complete instability of the $(n-1)$-dimensional boundary is sufficient.

1 Introduction

This paper is motivated by the important result of Kharitonov [1] which gives an elegant condition for the stability of a box of polynomials. We are interested in the converse of this problem; namely: *Can the complete instability of a box of polynomials be easily deduced?*

One motivation for studying this problem is the following. Consider a unity feedback system with a plant transfer function $\frac{N_p(s)}{D_p(s)}$ and a compensator chosen from the class

$$\left\{ \frac{N_c(s,\mathbf{u})}{D_c(s,\mathbf{v})} : \mathbf{u} \in U, \mathbf{v} \in V \right\}.$$

Here, the compensators are restricted to be of fixed structure; e.g., PID, observer-based or fixed order, with vectors $\mathbf{u}$ and $\mathbf{v}$ playing the role of design parameters. The objective is to find a compensator from this class which stabilizes the feedback system. This amounts to finding parameters $\mathbf{u} \in U$ and $\mathbf{v} \in V$ such that

$$D_c(s,\mathbf{v})D_p(s) + N_c(s,\mathbf{u})N_p(s) = 0 \tag{1}$$

has only roots in the open left half of the complex plane. Assuming polytopic U and V and affine dependence on $\mathbf{u}$ and $\mathbf{v}$, this stabilization problem is equivalent to determining whether a polytope of polynomials contains only unstable members. In this paper we study this problem for the special but important case of a box of polynomials and *show that a corresponding "Kharitonov" like result does not hold.* Specifically, we show that a box of polynomials may contain stable members even though its $(n-4)$-dimensional boundary is completely unstable and prove that it is sufficient to check the $(n-1)$-dimensional boundary.

2 Main Result

Consider an n^{th} order monic polynomial

$$f(s) = s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_n, \tag{2}$$

which we also represent as a vector $\mathbf{a} = (a_1, a_2, \cdots, a_n)$ in n-dimensional coefficient space. Let $H_n \subset \mathbf{R}^n$ denote polynomials with roots only in the open left half of the complex plane. If $\mathbf{a} \in H_n$ then we say the polynomial is stable, else it is unstable.

Definition 1: A subset T of the coefficient space is said to be *completely unstable* if the polynomial corresponding to each point in T is *unstable*.

Consider an n-dimensional box B_n in coefficient space given by

$$B_n = \{\mathbf{b} : b_i \in [\underline{b}_i, \overline{b}_i],\ \underline{b}_i > 0,\ i = 1, 2, \cdots, n.\} \tag{3}$$

Let ∂B_n^m denote the m-dimensional boundary of B_n; i.e., $\mathbf{b} \in \partial B_n^m$ if for $\{i_1, i_2, \cdots, i_n\} = \{1, 2, \cdots n\}$,

$$b_{i_1} \in \{\underline{b}_{i_1}, \overline{b}_{i_1}\}, \cdots, b_{i_{n-m}} \in \{\underline{b}_{i_{n-m}}, \overline{b}_{i_{n-m}}\}$$

and

$$b_{i_{n-m+1}} \in [\underline{b}_{i_{n-m+1}}, \overline{b}_{i_{n-m+1}}], \cdots, b_{i_n} \in [\underline{b}_{i_n}, \overline{b}_{i_n}].$$

Said another way, ∂B_n^m contains all the points generated by fixing $(n-m)$ coefficients at their extreme values and letting the remaining m coefficients vary. We now state our main result. The proof is given in the next section.

Theorem 1: *Consider an n-dimensional box B_n in coefficient space. If ∂B_n^{n-1} is completely unstable, then so is B_n. Moreover, the complete instability of ∂B_n^{n-4} does not imply the complete instability of B_n.*

3 Proof of Theorem 1

This section is divided into two parts. In the first subsection we derive supporting lemmas needed in the proof of the Theorem 1. In the second subsection we prove Theorem 1.

3.1 Supporting Lemmas

We first introduce some notation and definitions. For $\mathbf{a} = (a_1, a_2, \cdots, a_n) \in H_n$ and for the $n-m$ integers $\{i_1, i_2, \cdots, i_{n-m}\} \subset \{1, 2, \cdots, n\}$, let $H_n^m(\mathbf{a}; a_{i_1}, a_{i_2}, \cdots, a_{i_{n-m}}) \subset H_n$ denote the m-dimensional subset of Hurwitz polynomials which contains both $\mathbf{a}$ and all the polynomials with the same coefficients $a_{i_1}, a_{i_2}, \cdots, a_{i_{n-m}}$. Thus, all polynomials in $H_n^m(\mathbf{a}; a_{i_1}, a_{i_2}, \cdots, a_{i_{n-m}})$ have the same terms $a_{i_1} s^{n-i_1}, a_{i_2} s^{n-i_2}, \cdots, a_{i_{n-m}} s^{n-i_{n-m}}$. Let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ where α_i, $i = 1, 2, \cdots, n$ are the roots of $f(s) = 0$. We denote a sequence by $\{\mathbf{p}^k\}$ and its range as the set of points $\mathbf{p}^k, k = 1, 2, \cdots$. Since $\{\mathbf{p}^k\}$ is a vector sequence, the sequence of the i^{th} components is denoted by $\{p_i^k\}$. We denote the k^{th} terms of the sequences $\{\mathbf{p}^k\}$ and $\{p_i^k\}$ by $\mathbf{p}^k$ and p_i^k respectively.

Definition 2: A set $T \subseteq \mathbf{C}^n$ is said to be *unbounded* if for a given $\mathbf{q} \in \mathbf{C}^n$ there exists $\mathbf{p} \in T$ such that

$$\|\mathbf{p} - \mathbf{q}\| \geq M$$

for all $M \geq 0$.

Definition 3: A sequence $\{\mathbf{p}^k\}$ in $\mathbf{C}^n$ is *bounded (unbounded)* if its range is bounded (unbounded).

Lemma 1: *Consider some $T \subseteq \mathbf{C}^n$. The set T is unbounded if and only if there exists an unbounded sequence $\{\mathbf{p}^k\}$ in T.*

Proof: Proof follows from Definitions 2 and 3.

□

Lemma 2: *Consider a sequence $\{\boldsymbol{\alpha}^k\}$ in $\mathbf{C}^n$ and its associated sequence $\{\mathbf{a}^k\}$ in $\mathbf{R}^n$. Then, $\{\boldsymbol{\alpha}^k\}$ is unbounded if and only if $\{\mathbf{a}^k\}$ is unbounded.*

Proof:

- Sufficiency

  Assume $\{\mathbf{a}^k\}$ is unbounded. We need to show that $\{\boldsymbol{\alpha}^k\}$ is unbounded. Proceeding by contradiction assume that $\{\boldsymbol{\alpha}^k\}$ is bounded. This implies that all the n sequences $\{\alpha_i^k\}$, $i = 1, 2, \cdots, n$ in $\mathbf{C}$ are bounded. It is well known that the coefficients of a polynomial are multilinear functions of its roots. Hence, for $i \in \{1, 2, \cdots n\}$, a_i^k is a multilinear function of $\alpha_1^k, \alpha_2^k, \cdots, \alpha_n^k$. Thus, it follows that the sequences $\{a_i^k\}$ in $\mathbf{R}$ are bounded. This implies that $\{\mathbf{a}^k\}$ is bounded which is a contradiction. Hence $\{\boldsymbol{\alpha}^k\}$ is unbounded.

- Necessity

  Assume $\{\boldsymbol{\alpha}^k\}$ is unbounded. We need to show that $\{\mathbf{a}^k\}$ is unbounded. Again, proceeding by contradiction, assume $\{\mathbf{a}^k\}$ is bounded. This implies that all the n sequences $\{a_i^k\}$, $i = 1, 2, \cdots, n$ in $\mathbf{R}$ are bounded. Without loss of generality assume that the sequence $\{\alpha_1^k\}$ in $\mathbf{C}$ is unbounded. By definition, α_1^k is a root of the polynomial in (2). Hence, it follows that

  $$(\alpha_1^k)^n + a_1^k(\alpha_1^k)^{n-1} + \cdots + a_n^k = 0. \tag{4}$$

  Because $\{\alpha_1^k\}$ is unbounded, there exist some K such that $|\alpha_1^k| > 0$ for all $k > K$. Thus, for sufficiently large K, (4) can be written as

  $$a_1^k \frac{(\overline{\alpha}_1^k)}{|\alpha_1^k|^2} + \cdots + a_n^k \frac{(\overline{\alpha}_1^k)^n}{|\alpha_1^k|^{2n}} = -1. \tag{5}$$

By assumption, $\{\alpha_1^k\}$ is unbounded while $\{a_i^k\}$, $i = 1, 2, \cdots, n$ are bounded. It thus follows that

$$\lim_{k \longrightarrow \infty} a_i^k \frac{(\overline{\alpha}_1^k)^i}{|\alpha_1^k|^{2i}} = 0, \qquad i = 1, 2, \cdots, n.$$

Since these individual limits exists, we can now write

$$\lim_{k \longrightarrow \infty} (a_1^k \frac{(\overline{\alpha}_1^k)}{|\alpha_1^k|^2} + \cdots + a_n^k \frac{(\overline{\alpha}_1^k)^n}{|\alpha_1^k|^{2n}}) = 0. \tag{6}$$

However, from (5)

$$\lim_{k \longrightarrow \infty} (a_1^k \frac{(\overline{\alpha}_1^k)}{|\alpha_1^k|^2} + \cdots + a_n^k \frac{(\overline{\alpha}_1^k)^n}{|\alpha_1^k|^{2n}}) = -1. \tag{7}$$

Clearly, equations (6) and (7) are in contradiction. Hence $\{\mathbf{a}^k\}$ is unbounded.

□

In the sequel we will repeatedly use the fact that for $i \in \{1, 2, \cdots, n\}$ the roots and coefficients of a polynomial $f(s)$ are related by the equation

$$(-1)^i a_i = \Sigma(i) \tag{8}$$

where $\Sigma(i)$ denotes the sum of the products of the roots α_h, $h = 1, 2, \cdots, n$, taken i at a time. Similarly, let $\Sigma(i; l_1, \cdots, l_m)$ denote the sum of the product of roots α_h, $h = 1, 2, \cdots, n$, $h \neq l_1, \cdots, l_m$, taken i at a time. In the proof of our next lemma, we will use the fact that for given $\tilde{a} > 0$ and $i \in \{1, 2, \cdots, n\}$ there exists a polynomial $f(s)$ having only negative roots such that its i^{th} coefficient a_i satisfies $a_i = \tilde{a}$. To see this fact, use (8) and write

$$\Sigma(i) = \alpha_q \Sigma(i-1; q) + \Sigma(i; q) \tag{9}$$

where $q \in \{1, 2, \cdots, n\}$. Rearranging terms gives

$$\alpha_q = \frac{\Sigma(i) - \Sigma(i; q)}{\Sigma(i-1; q)}. \tag{10}$$

We now choose the numbers $\alpha_1, \cdots, \alpha_{q-1}, \alpha_{q+1}, \cdots, \alpha_n$ to be negative such that $|\Sigma(i; q)| < |\Sigma(i)| = \tilde{a}$. It then follows that α_q obtained from (10) is a negative real number since $sgn(\Sigma(i)) = -sgn(\Sigma(i-1; q))$.

Lemma 3: *For* $\mathbf{a} \in H_n$ *and* $i \in \{1, 2, \cdots, n\}$, $H_n^{n-1}(\mathbf{a}; a_i)$ *is unbounded.*

Proof: From our preceding argument there always exists a point in $H_n^{n-1}(\mathbf{a}; a_i)$ whose corresponding polynomial has only negative real roots. Without loss of generality we assume $\mathbf{a}$ is such a polynomial. We now consider two cases.

- **Case 1** ($i = 1$):
  We will show that the (n-1)-dimensional set $H_n^{n-1}(\mathbf{a}; a_1)$ is unbounded. Let $\boldsymbol{\alpha} \in \mathbf{C}^n$ be the vector of roots of the polynomial corresponding to $\mathbf{a} \in H_n$. From (8),

$$a_1 = -\sum_{i=1}^{n} \alpha_i.$$

  Since $a_i \in \mathbf{R}$ for all i, we have

$$a_1 = -\sum_{q=1}^{n} Re(\alpha_q). \tag{11}$$

  Select $l, h \in \{1, 2, \cdots, n\}$. We shall now construct an unbounded sequence of roots $\{\boldsymbol{\alpha}^k\}$. To begin, let $\boldsymbol{\alpha}^1 = \boldsymbol{\alpha}$ and for $k = 2, 3, \cdots$, take

$$\begin{aligned} \alpha_i^k &= \alpha_i, \qquad i \neq l, h; \\ \alpha_l^k &= \frac{1}{2} Re(\alpha_l + \alpha_h) + jk; \\ \alpha_h^k &= \overline{\alpha}_l^k. \end{aligned} \tag{12}$$

  Since the imaginary parts of α_h^k and α_l^k are unbounded, then the sequences $\{\alpha_l^k\}$ and $\{\alpha_h^k\}$ are unbounded in $\mathbf{C}$. This implies that the sequence $\{\boldsymbol{\alpha}^k\}$ is unbounded in $\mathbf{C}^n$. Let the corresponding sequence of coefficient vectors be denoted by $\{\mathbf{a}^k\}$ with $\mathbf{a}^1 = \mathbf{a}$. From Lemma 2 we see that $\{\mathbf{a}^k\}$ is also unbounded. It follows from (11) and (12) that $a_1^k = a_1$ for all k. Thus, $\mathbf{a}^k \in H_n^{n-1}(\mathbf{a}; a_1)$ for all k. Using Lemma 1 we conclude that $H_n^{n-1}(\mathbf{a}; a_1)$ is unbounded.

- **Case 2** ($H_n^{n-1}(\mathbf{a}; a_i)$, $i \neq 1$):
  Form a sequence $\{\boldsymbol{\alpha}^k\}$ with $\boldsymbol{\alpha}^1 = \boldsymbol{\alpha}$ such that

$$\alpha_h^k = \frac{\alpha_h}{k} \qquad h = 2, 3, \cdots, n \;\; h \neq l, \quad k = 2, 3, \cdots \tag{13}$$

  and where α_l^k is computed using (10). Because of this, $a_i^k = a_i$ for all k. Hence, $\{\alpha_l^k\}$ is an unbounded sequence in $\mathbf{C}$ which implies that $\{\boldsymbol{\alpha}^k\}$ is unbounded. Let the corresponding sequence of coefficient vectors be denoted by $\{\mathbf{a}^k\}$ with $\mathbf{a}^1 = \mathbf{a}$. From Lemma 2 we see that $\{\mathbf{a}^k\}$ is unbounded. Using Lemma 1 we conclude that $H_n^{n-1}(\mathbf{a}; a_i)$, $i \neq 1$ is unbounded. This completes the proof of Case 2.

□

Lemma 4: *Consider an unbounded sequence* $\{\mathbf{a}^k\}$ *in* $H_n^{n-1}(\mathbf{a}; a_2)$ *and the corresponding sequence* $\{\boldsymbol{\alpha}^k\}$ *in* $\mathbf{C^n}$. *Then, only one sequence amongst* $\{\alpha_i^k\}, i = 1, 2, \cdots, n$ *is unbounded. Moreover, for this sequence, say* $\{\alpha_q^k\}$ *where* $q \in \{1, 2, \cdots, n\}$, *there exists a* $K > 0$ *such that* $Im(\alpha_q^k) = 0$ *for all* $k \geq K$.

Proof: From Lemma 2, $\{\mathbf{a}^k\}$ unbounded implies $\{\boldsymbol{\alpha}^k\}$ is unbounded. Hence, it is necessary that at least one of the n sequences $\{\alpha_i^k\}, i = 1, 2, \cdots, n$ is unbounded. We first prove that only one component root sequence amongst $\{\alpha_i^k\}$, $i = 1, 2, \cdots, n$ is unbounded. Proceeding by contradiction, assume that for $j > 1$ and $\{i_1, i_2, \cdots, i_j\} \subset \{1, 2, \cdots, n\}$ that the sequences $\{\alpha_{i_1}^k\}, \cdots, \{\alpha_{i_j}^k\}$ are unbounded. From these j unbounded component root sequences select two real sequences or a complex conjugate pair sequence. Without loss of generality assume they are $\alpha_{i_1}^k$ and $\alpha_{i_2}^k$. From (8) and (9) we obtain

$$a_2 = a_2^k = \alpha_{i_1}^k \alpha_{i_2}^k + (\alpha_{i_1}^k + \alpha_{i_2}^k)\Sigma(1; i_1, i_2) + \Sigma(2; i_1, i_2). \tag{14}$$

In right hand side of the above, all three terms are positive and $\alpha_{i_1}^k \alpha_{i_2}^k$ becomes unbounded as $k \longrightarrow \infty$. Hence, $\{a_2^k\}$ is unbounded which is a contradiction. Thus, there exists only one unbounded sequence amongst $\{\alpha_i^k\}, i = 1, 2, \cdots, n$.

We now have to prove that the unbounded sequence $\{\alpha_q^k\}$ eventually becomes real. Proceeding by contradiction, assume that there exists a $K > 0$ such that $Im(\alpha_q^k) \neq 0$ for all $k > K$. This implies that there are two unbounded sequences $\{\alpha_q^k\}$ and $\{\overline{\alpha}_q^k\}$ which contradicts the first result.

□

Lemma 5: *Consider an unbounded sequence $\{\mathbf{a}^k\}$ in $H_n^{n-1}(\mathbf{a}; a_1)$ and the corresponding sequence $\{\boldsymbol{\alpha}^k\}$ in $\mathbf{C}^n$. Then,*

$$|Re(\alpha_i^k)| < a_1 \tag{15}$$

for all $i = 1, 2, \cdots, n$ and $k = 1, 2, \cdots$.

Proof: Expressing a_1 in terms of roots gives

$$a_1 = a_1^k = -\sum_{i=1}^{n} \alpha_i^k = -\sum_{i=1}^{n} Re(\alpha_i^k) \tag{16}$$

Because $\mathbf{a}^k \in H_n^{n-1}(\mathbf{a}; a_1)$, for all k

$$Re(\alpha_i^k) < 0, \qquad i = 1, 2, \cdots, n, \quad k = 1, 2, \cdots. \tag{17}$$

Inequality (15) follows from (16) and (17).

□

Lemma 6: *$H_n^{n-2}(\mathbf{a}; a_1, a_2)$ is bounded.*

Proof: It can be easily seen that

$$H_n^{n-2}(\mathbf{a}; a_1, a_2) = H_n^{n-1}(\mathbf{a}; a_1) \cap H_n^{n-1}(\mathbf{a}; a_2).$$

Consider a sequence $\{\mathbf{a}^k\}$ in $H_n^{n-2}(\mathbf{a}; a_1, a_2)$ and the corresponding root sequence $\{\boldsymbol{\alpha}^k\}$ in $\mathbf{C}^n$. From Lemma 4 it follows that there exists an $M_1 > 0$ such that $|Im(\alpha_i^k)| < M_1$ for all $i \in \{1, 2, \cdots, n\}$ and $k = 1, 2, \cdots$. From Lemma 5 it follows that there exists an $M_2 > 0$ such that $|Re(\alpha_i^k)| < M_2$ for all $i \in \{1, 2, \cdots, n\}$ and $k = 1, 2, \cdots$. Hence, $\{\boldsymbol{\alpha}^k\}$ is bounded which implies from Lemma 2 that $\{\mathbf{a}^k\}$ is bounded. Since the choice of $\{\mathbf{a}^k\}$ was arbitrary it follows from Lemma 1 that $H_n^{n-2}(\mathbf{a}; a_1, a_2)$ is bounded.

□

Lemma 7: *Consider $\{\mathbf{a}^k\}$ in H_n and the corresponding $\{\alpha^k\}$ in $\mathbf{C}^n$. If $|\alpha_i^k| > m_1 > 0$ for all $i = 1, 2, \cdots, n$ and $k = 1, 2, \cdots$, then, there exists an $m > 0$ such that $a_i^k > m$ for $i = 2, 3, \cdots, n$ and $k = 1, 2, \cdots$.*

Proof: Given $i \in \{1, 2, \cdots, n\}$, let $\Sigma^k(i)$ be the sum of the product of the roots α_h^k, $h = 1, 2, \cdots, n$, taken i at a time; in other words, $\Sigma^k(i) = (-1)^i a_i^k$. Separate the terms in $\Sigma^k(i)$ into the two groups

$$\Sigma^k(i) = g_1^k + g_2^k \tag{18}$$

where the first group g_1^k consists only of those terms which contain at least one complex root without its complex conjugate. All terms which do not satisfy this property are collected in g_2^k. Because g_2^k only contains negative real numbers and complex conjugate pairs, the sign of every term in g_2^k is $(-1)^i$. Let the number of terms in g_2^k be l and sum of all terms in g_2^k be denoted by $\sum g_2^k$. It then follows that

$$(-1)^i \sum g_2^k > l m_1^i > 0. \tag{19}$$

A straightforward analysis shows that $sgn(g_2^k) = sgn(g_1^k) = (-1)^i$. Thus, it follows from (18)-(19) that

$$a_i^k = (-1)^i (g_1^k + g_2^k) > l m_1^i > 0, \quad i = 2, 3, \cdots, n.$$

The proof is completed by taking $m = l m_1^i$.

□

Lemma 8: *For any $\mathbf{p} = (p_1, p_2, \cdots, p_n) \in H_n^{n-3}(\mathbf{a}; a_1, a_2, a_n)$, there exists M and m such that $0 < m < p_i < M < \infty$, for $i = 1, 2, \cdots, n$.*

Proof: From Lemma 6 we know that $H_n^{n-2}(\mathbf{a}; a_1, a_2)$ is bounded. This implies that $H_n^{n-3}(\mathbf{a}; a_1, a_2, a_n)$ is bounded, because $H_n^{n-3}(\mathbf{a}; a_1, a_2, a_n) \subset H_n^{n-2}(\mathbf{a}; a_1, a_2)$. Hence, it follows that there exists an M such that,

$$p_i < M < \infty, \; i = 1, 2, \cdots, n.$$

Consider a sequence $\{\mathbf{p}^k\}$ in $H_n^{n-3}(\mathbf{a}; a_1, a_2, a_n)$ and the corresponding root sequence $\{\alpha^k\}$. By definition, every polynomial in $H_n^{n-3}(\mathbf{a}; a_1, a_2, a_n)$ has the same constant term a_n. Thus,

$$a_n = p_n = p_n^k = (-1)^n \alpha_1^k \, \alpha_2^k \cdots \alpha_n^k, \quad k = 1, 2, \cdots. \tag{20}$$

We claim that there exists an m_1 such that

$$|\alpha_i^k| > m_1 > 0, \; i = 1, 2, \cdots, n, \;\; k = 1, 2, \cdots.$$

This can be seen as follows. Proceeding by contradiction, assume that there exists no such $m_1 > 0$. This implies that there exists a sequence $\{\alpha_i^k\}$ such that

$$\lim_{k \longrightarrow \infty} \alpha_i^k = 0.$$

Then, for (20) to hold, there must be at least one unbounded sequence $\{\alpha_h^k\}$, $h \neq i$. But this implies that $H_n^{n-2}(\mathbf{a}; a_1, a_2, a_n)$ is unbounded, which is a contradiction. Thus, such a number $m_1 > 0$ exists. It then follows from Lemma 7 that there exists an m such that

$$p_i > m > 0, \quad i = 1, 2, \cdots, n.$$

The proof is completed.

□

Lemma 9: *H_n is connected and unbounded.*

Proof: Let $D_n \subset \mathbf{R}^n$ denote polynomials with roots only in the open unit disk. It was shown in [2] that D_n is a connected region. Since H_n can be mapped to D_n under a bilinear transformation, then H_n is also connected. Unboundedness of H_n follows easily from Lemma 3 which shows that a particular subset of H_n is unbounded.

□

Lemma 10: *B_n is completely unstable if and only if ∂B_n^{n-1} is completely unstable.*

Proof: Necessity is obvious. To prove sufficiency assume that ∂B_n^{n-1} is unstable. We have to show that B_n is unstable. We have proved in Lemma 9 that H_n is connected and unbounded. Proceeding by contradiction, assume that there exists a point $\mathbf{a}$ in the interior of B_n such that the polynomial corresponding to $\mathbf{a}$ is stable. From the unboundedness of H_n and the boundedness of B_n we conclude that H_n can not be wholly contained in B_n. Then, from the connectedness of H_n, it follows that some point on ∂B_n^{n-1} must be stable. This is a contradiction. Hence, B_n is completely unstable.

□

3.2 Proof of Theorem 1

We have already shown in Lemma 10 that the complete instability of ∂B_n^{n-1} implies that B_n is completely unstable. To complete the proof, we proceed by contradiction and assume that ∂B_n^{n-4} being completely unstable is sufficient for B_n to be completely unstable. Given $\mathbf{a} \in H_n$ we have shown in Lemma 8 that if $\mathbf{p} = (p_1, p_2, \cdots, p_n) \in H_n^{n-3}(\mathbf{a}; a_1, a_2, a_n)$, then $0 < m < p_i < M < \infty$. Now, consider the box B_n

$$B_n = \{\mathbf{b} : b_i \in [\underline{b}_i, \overline{b}_i]\} \tag{21}$$

such that $\underline{b}_i = m$, $\overline{b}_i = M$, $i = 3, 4, \cdots, (n-1)$ and $\underline{b}_i = a_i$, $\overline{b}_i = M$, $i = 1, 2, n$. It is easy to see that $H_n^{n-3}(\mathbf{a}; a_1, a_2, a_n)$ is contained in the $(n-3)$-dimensional boundary of B_n obtained by fixing $b_1 = \underline{b}_1, b_2 = \underline{b}_2$ and $b_n = \underline{b}_n$. This shows that complete instability of ∂B_n^{n-4} is not enough to conclude that B_n is completely unstable, which is a contradiction.

□

4 Conclusions

To determine whether B_n is completely unstable we have shown that at least the $(n-3)$-dimensional boundary of B_n and, at most, its $(n-1)$-dimensional boundary must be checked. However, we believe that a tighter result is possible; i.e., that B_n is completely unstable if and only if the $(n-2)$-dimensional boundary is completely unstable. We save this possibility for future investigation.

Acknowledgements

This work was supported by the National Science Foundation under Grant NSF-8858366.

References

[1] V.L. Kharitonov, Asymptotic Stability of an Equilibrium Position of a Family of Systems of Linear Differential Equations, *Differential'nye Uraveniya,* Vol.14, pp. 2086-2088, 1978.

[2] A.T. Fam and J.S. Meditch, A Canonical Parameter Space for Linear Systems Design, *IEEE Transactions on Automatic Control*, Vol. AC-23, pp. 454-458, 1978.

On H_∞ Optimal Structural Variations

David Bensoussan Maher Hammami
École de technologie supérieure
Université du Québec
Département de génie électrique
4750, Henri-Julien, Montréal (Québec)
Canada H2T 2C8

Abstract

In this paper an alternative approach to the control of plants having an extra degree of freedom - the variable structure - is elaborated [1]. The plant variation can be manipulated so as to achieve an improved performance objective and is shown to be the solution of a standard H_∞ problem. Moreover, the control can be further improved by feedforwarding disturbance estimates.

Introduction

The recent developments in H_∞ control theory [2] provide an accurate framework of reference which sometimes lead to quite conservative results. Since the H_∞ weighted norm depends on the supremum over all frequencies of the maximum singular value, H_∞ control remains an elegant but worst case design solution i.e. a conservative one. Such conservativeness can be reduced whenever the plant includes some added structural variation which can be initiated and measured. The process can be further improved when the disturbances have a waveform structure, i.e. their frequency responses can be tracked and measured (eg. wind gusts). In such a case, well proven predictive estimations [3] can be used to better improve the control process.

This paper restricts itself to the particular case of plants having an extra degree of freedom embedded in a particular structural variation which can be initiated and measured. The manoeuvres of a sailboat typically exemplify such plants: structural variations consist of reorientation or deployment of sails as well as modification of the center of gravity. Moreover, the sailboat adapts itself to changing winds so as to use positively the wind energy according to performance objectives. Such an example motivates the present analysis and clearly defines the framework of the following developments.

Optimization of the Variable Structure

Let P be the plant, C the compensator, y the plant output, v the plant input, r the reference, e the error signal between reference and output, d_1 the additive output disturbances, d_2 the sensor noise and d_3 the plant input noise [Fig. 1]. The feedback system can be represented in the standard H_∞ form in which w represents the variables not accessible to the controller, z the regulator variables not accessible to the controller, y the accessible outputs and u the control variable which can be manipulated by the controller K of the plant G [Fig. 2]. It follows that:

$$z = [G_{11} + G_{12} K \ (I - G_{22} K)^{-1} G_{21}] \ w$$
$$y = (I - G_{22} K)^{-1} G_{21} \ w$$

The variable structure ΔP (or similarily $C\Delta P$) can be shown to be the solution of a standard H_∞ problem in which:

$$w = \begin{pmatrix} I \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} \qquad z = e \qquad y = v \qquad u = Kv$$

$$G_{11} = (I + QPC,\ Q,\ Q,\ QP) \qquad G_{12} = Q$$

$$G_{21} = [C + CQPC,\ CQ,\ CQ,\ I + CQP) \qquad G_{22} = CQ$$

$$\text{with } Q = -(I + PC)^{-1}, \qquad K = \Delta P$$

Although attaining ΔP is obviously limited in practice, knowing ΔP of gives an interesting insight at the design stage of the variable structure, so as to achieve better performance e.g. improving bounds in the mixed sensitivity problem.

Feedforwarding disturbance estimates

We now assume that the disturbances d_1, d_2 and d_3 can be estimated to $\hat{d}_1$,$\hat{d}_2$ and $\hat{d}_3$. In such a case, the new standard problem can be rewritten as follows:

$$w = \begin{pmatrix} r \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} \qquad z = \begin{pmatrix} e \\ v \end{pmatrix} \qquad y = \begin{pmatrix} y \\ \hat{d}_1 \\ \hat{d}_2 \\ \hat{d}_3 \end{pmatrix} = \begin{pmatrix} y \\ \hat{d} \end{pmatrix}, \qquad u = Ky$$

$$K = (-I,\ -J) \quad \text{and}$$

$$z = [G_{11} + G_{12}\, K\, (I - G_{22}\, K)^{-1}\, G_{21}]\ w - G_{12}\ (I - G_{22}\, K)^{-1}\, J\, \hat{d}$$

$$y = (I - G_{22}\, K)^{-1}\, G_{21}\ w - (I - G_{22}\, K)^{-1}\, G_{22}\, J\, \hat{d}$$

The choice of J allows a better tuning of the control process according to performance objectives.

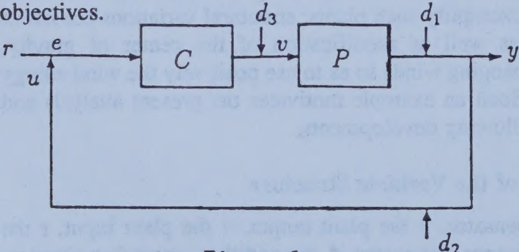


Figure 1

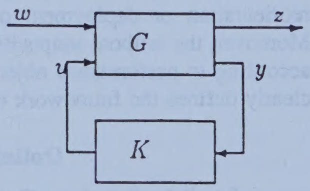


Figure 2

Conclusion

The combination of both structure variations and feedforwarding of disturbance estimates can further help towards achieving desirable weighted norms on the transmissions of the feedback systems. Disturbance effects can then be beneficially used in conjunction with the flexible structure in the plant (e.g. sailboat advancing into the wind).

[1] Bensoussan, D., "*A New Feedback Optimization Problem*", Proceedings of the 28th IEEE Conference on Decision and Control, Tempa, Florida, December 1989, p. 2375.

[2] "*Robust Control*", Edited by Peter Dorato, IEEE Press 1987.

[3] Johnson, C.D., "*Theory of Disturbance Accomodating Controllers*", Advances in Control and Dynamical Systems, Vol. 12, (C.T. Leondes ed.), Academic Press, New York, 1976.

Characteristics of Riccati Solutions in a General $\mathcal{H}_\infty$ Control Problem

Pradeep Pandey [*]
Dept. Elec. Comp. Engg.
Univ. of California
Santa Barbara, CA 93106

Andy Packard [†]
Dept. Mechanical Engg.
Univ. of California
Berkeley, CA 94720

Alan J. Laub [†]
Dept. Elec. Comp. Engg.
Univ. of California
Santa Barbara, CA 93106

1 Introduction

Consider an $\mathcal{H}_\infty$ control problem where we want to find a controller that is internally stabilizing and minimizes a closed-loop $\mathcal{H}_\infty$ norm. In [1, 2] the authors provided a state-space approach to finding γ_{opt}, the optimal gain, that relies on formulating the problem in terms of two Riccati equations which depend on γ. The optimal gain is the infimum over all suboptimal gains, γ, such that the spectral radius of the product of the Riccati solutions is bounded by γ^2.

In this paper we characterize the behavior of the Riccati solutions via their dependence on γ. These Riccati equations depend on γ in a rather complicated way when the disturbance feedthrough matrix, D_{11}, is not zero. It was noted in [2] that a given problem can be transformed to an equivalent one where $D_{11} = 0$. However, the transformations involved depend on the value of γ under consideration. This makes it harder to characterize the Riccati solutions as a *function* of γ. For this reason we define a *general* $\mathcal{H}_\infty$ problem to be one that satisfies all the assumptions from [1] except that $D_{11} \neq 0$. Our aim is to characterize the Riccati solutions in a general problem.

2 Characterizing the Riccati Solutions

Consider the following linear system with two sets of multivariable inputs and outputs:

$$\dot{x} = Ax + B_1 w + B_2 u, \quad z = C_1 x + D_{11} w + D_{12} u, \quad y = C_2 x + D_{21} w + D_{22} u,$$

which satisfies the assumptions of a general problem. We first define the Hamiltonian associated with this problem. For notational simplicity we define $\alpha := \gamma^{-2}$. Scalar- and matrix-valued functions of α will be subscripted with α. If f is such a function then f' and f'' will denote its first and second derivative, respectively, with respect to α.

Let $D_H := [\; D_{11} \quad D_{12} \;]$ and $M := [\; D_H^T C_1 \quad B^T \;]^T$, and define the Hamiltonian H_α by

$$H_\alpha := \begin{bmatrix} A & 0 \\ -C_1^T C_1 & -A^T \end{bmatrix} - \mathcal{J} M^T R_\alpha^{-1} M =: \begin{bmatrix} A_\alpha & -F_\alpha \\ -G_\alpha & -A_\alpha^T \end{bmatrix}, \tag{1}$$

where

$$\mathcal{J} := \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}, \quad R_\alpha := D_H^T D_H - \begin{bmatrix} \alpha^{-1} I & 0 \\ 0 & 0 \end{bmatrix}. \tag{2}$$

A dual Hamiltonian J_α is defined similarly. In the sequel we describe the behavior of H_α and X_α. By duality, these comments also apply to J_α and Y_α. Consider the Riccati equation associated with H_α in (2)

$$\mathcal{R}(X, \alpha) := A_\alpha^T X_\alpha + X_\alpha A_\alpha - X_\alpha F_\alpha X_\alpha + G_\alpha = 0. \tag{3}$$

By an *admissible* solution we will mean a unique nonnegative definite stabilizing solution. For $\alpha = 0$ our assumptions guarantee *a priori* that an admissible $X_0 := Ric(H_0) \geq 0$ exists. However, for $\alpha > 0$ it is not clear for what values of α an admissible solution will exist. By continuity, H_α will be in $dom(Ric)$ for some $\alpha > 0$. Suppose H_α and J_α first fail to be in $dom(Ric)$ at α_x and α_y, respectively, and define $\alpha_* := \min(\alpha_x, \alpha_y)$. We will refer to an $\alpha \in (0, \alpha_*)$ as *feasible.*

---

[*] Supported in part by National Science Foundation (and AFOSR) under Grant No. ECS 87-18897.

[†] Supported in part by the General Electric Company and the UC MICRO Program (89-096).

Partition D_{11} as $\begin{bmatrix} D_{1111} & D_{1112} \\ D_{1121} & D_{1122} \end{bmatrix}$, in accordance with the dimensions of B_1, B_2, C_1, and C_2, and define σ_d by

$$\sigma_d := \max\Big(\ \sigma_{max}[\ D_{1111}\ \ D_{1112}\ ]\ ,\ \sigma_{max}[\ D_{1111}^T\ \ D_{1121}^T\ ]^T\Big).$$

As noted in [2], by considering the general problem at $\omega = +\infty$ we conclude that $\gamma_{opt} \geq \sigma_d$. Hence, to find α_{opt} we need only search the interval $(0, \sigma_d^{-2})$. For this range of α we find that G_α remains nonnegative definite and we can write

$$A_\alpha = A - \widetilde{A}_\alpha C_1, \quad \text{and} \quad G_\alpha = C_\alpha^T C_\alpha \quad \text{where} \quad C_\alpha = \widetilde{C}_\alpha C_1$$

for some $\widetilde{A}_\alpha$ and $\widetilde{C}_\alpha$. This observation is crucial in checking if (A_α, C_α) is observable/detectable and is used in Theorems 1 and 2 below, which are proved in [3].

Theorem 1 *For feasible α we can write $G_\alpha = C_\alpha^T C_\alpha$. If $\mathcal{V}$ spans the unobservable subspace of the pair (A, C_1) then it also spans the unobservable subspace of (A_α, C_α). Further, $\ker(X_\alpha) = \mathcal{V}$.*

Theorem 2 *Let $\alpha_* > \alpha_1 > \alpha_0 \geq 0$. Then for a general problem the Riccati solutions X_α and Y_α, and the spectral radius function $\rho_\alpha := \rho(X_\alpha Y_\alpha)$ are continuous and nondecreasing, i.e., $X_{\alpha_1} \geq X_{\alpha_0}$, and $\rho_{\alpha_1} \geq \rho_{\alpha_0}$.*

Theorem 3 *For feasible α there exists an analytic function X_α which solves the Riccati equation $\mathcal{R}(X, \alpha) = 0$.*

Proof: Consider the Riccati equation (3) and take the derivative with respect to α to get

$$A_c^T X' + X' A_c - \mathcal{D} = 0, \tag{4}$$

where $\mathcal{D} := [\ I\ \ X_\alpha\ ]\mathcal{J}H' \begin{bmatrix} I \\ X_\alpha \end{bmatrix}$, and $A_c := A_\alpha - F_\alpha X_\alpha$. Further, the gradient $\mathcal{J}H'$ is

$$\mathcal{J}H' = -M^T R_\alpha^{-1} \begin{bmatrix} \alpha^{-2} I & 0 \\ 0 & 0 \end{bmatrix} R_\alpha^{-1} M \leq 0.$$

For feasible α the Riccati solution X_α is admissible implying that the closed-loop matrix A_c is stable. Hence, the closed-loop Lyapunov operator $\Omega(Z) := A_c^T Z + Z A_c$ is invertible and we have a unique solution $X' = \Omega^{-1}(\mathcal{D}) \geq 0$. In fact, Ω is the partial Fréchet derivative of $\mathcal{R}(X, \alpha)$ with respect to X, and its invertibility implies, via the implicit function theorem, the existence of an analytic function X_α. ∎

Hence, we have shown that the Riccati solutions X_α and Y_α are ordered and analytic, and the spectral radius $\rho(X_\alpha Y_\alpha)$ is nondecreasing. These results are used in [4] to compute γ_{opt}.

References

[1] J.C. Doyle, K. Glover, P. Khargonekar, and B.C. Francis, "State-space Solutions to Standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ Control Problems," *IEEE Trans. Auto. Cont.*, vol. AC-34, pp. 831–847, 1989.

[2] K. Glover and J. Doyle, "State-space Formulas for all Stabilizing Controllers that Satisfy an $\mathcal{H}_\infty$ Norm Bound and Relations to Risk Sensitivity," *Sys. and Cont. Letters*, vol. 11, pp. 167–172, 1988.

[3] P. Pandey, and A.J. Laub, "Computing the Optimal Norm for a General $\mathcal{H}_\infty$ Control Problem," Report SCL 90-03, Dept. Elec. Comp. Engg., Univ. California, Santa Barbara, August 1990.

[4] P. Pandey, C. Kenney, A. Packard, and A.J. Laub, "A Gradient Method for Computing the Optimal $\mathcal{H}_\infty$ Norm," to appear in *IEEE Trans. Auto. Cont.*, March 1991.

Schur-Stability of a Class of Parameter-Dependent Matrices

F. Garofalo L. Glielmo
L. Bevilacqua

Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"
via Claudio 21, 80125 Napoli, Italy

Abstract

In this paper a sufficient condition for the Schur-stability of a class of matrices is given. The matrices have elements which are multilinear functions of a set of uncertain, bounded, possibly time-varying parameters. The condition for stability requires the check for positive definiteness of a finite number of quadratic forms.

1 Introduction

The problem of robust stability of interval matrices has recently obtained an increased attention. The interested reader can refer to [1] where a concise overview of the results available on this topic is given.

An interval matrix is a matrix whose elements are known to vary independently within given bounds. The conditions for the stability of such kind of matrices can be usefully applied to check the stability of an uncertain plant whose dynamical matrix can be approximated by an interval matrix. Often, however, the assumption that the elements of a dynamical matrix of a plant vary independently of one another is unrealistic and the stability conditions obtainable under these hypothesis are too restrictive. A more realistic assumption would require that each element of the matrix be a function of a set of parameters which, in turn, vary within given bounds.

A linear dependence of the elements of the matrix from the set of parameters has been considered in [2], but again this kind of dependence is not suitable to model many real world situations. A better approximation can be obtained by considering a multilinear dependence of the elements of the matrix from the parameters. This case has been considered in [3]. One of the results therein is that Hurwitz-stability of the above mentioned class of uncertain matrices can be guaranteed by examining the positive definiteness of a quadratic forms in correspondence of a *finite number of points in the parameter space.* This fact is based on continuous time Lyapunov arguments and cannot be directly used to prove Schur-stability. To this aim, a generalization of the result in [3], recently presented in [4] and [5] for continuous time systems, can be used, as it will be shown in the sequel.

2 Problem Statement and Summary of Some Useful Results

We consider an uncertain system described by the following difference equation

$$x(k+1) = A[p(k)]x(k), \tag{1}$$

where $x(k) \in \mathbf{R}^n$ denotes the state of the system, and $p : Z \to \wp \subset R^\nu$ is a time-varying vector of uncertain parameters. The *a priori* knowledge about the uncertainty affecting the system, concerns the set $\wp$ which is assumed to be the hyperrectangle[1]

$$\wp \triangleq \{p \in \mathbf{R}^\nu | 0 \leq \underline{p} \leq p \leq \overline{p}\}, \tag{2}$$

where $\underline{p}, \overline{p} \in \mathbf{R}^\nu$ are known vectors and the inequalities are intended component-wise.

The matrix function $A : p \mapsto A(p)$ is *multilinear* in the components of p, i.e.

$$A(p) = \sum_{(i_1,\ldots,i_\nu)\in \mathcal{B}^\nu} A_{i_1,\ldots,i_\nu} p_1^{i_1} \cdots p_\nu^{i_\nu}, \tag{3}$$

where p_i are the components of vector p, $A_{i_1,i_2,\ldots,i_\nu}$ are constant $n \times n$ matrices, $\mathcal{B} \triangleq \{0,1\}$, and $\mathcal{B}^\nu \triangleq \mathcal{B} \times \cdots \times \mathcal{B}$ (ν times).

Regarding the stability of system (1) we make the assumption that there exists a constant *nominal* value of the parameters, $p_0 \in \wp$, for which matrix $A_0 \triangleq A(p_0)$ is Schur-stable. This is equivalent to say that for any symmetric positive definite matrix Q_0, there exists a unique symmetric positive definite matrix P_0 such that [6]

$$A_0^T P_0 A_0 - P_0 = -Q_0. \tag{4}$$

We now focus our attention on the problem of checking the stability of system (1) for any $\wp$-valued function $p(\cdot)$. On the basis of Lyapunov's second method for discrete-time systems [6], a sufficient condition is that the matrix

$$Q(p) \triangleq -A^T(p) P_0 A(p) + P_0 \tag{5}$$

be positive definite on $\wp$.[2]

In general it is not possible to check this condition in a simple way, but the information available about the structure of the matrix $A(p)$ (and hence of $Q(p)$) can help in reducing it to a tractable one. We can cite, in this respect, a result in [7] that we summarize below.

Fact 1 *Let $\underline{a}_{ij}$ and $\overline{a}_{ij}$ be the minimum and the maximum over the set $\wp$ of each element $a_{ij}(\cdot)$ of the matrix $A(\cdot)$.*[3] *Let $Q^{[h]} \triangleq -A^{[h]T} P_0 A^{[h]} + P_0$, where $A^{[h]} \triangleq \{a_{ij}^{[h]}\}$, $h = 1, \ldots, 2^{n^2}$, and*

$$a_{ij}^{[h]} = \underline{a}_{ij} \quad \text{or} \quad a_{ij}^{[h]} = \overline{a}_{ij}, \quad i,j = 1, \ldots, n. \tag{6}$$

Then, a sufficient condition for matrix $Q(p)$ to be positive definite on $\wp$ is that matrices $Q^{[h]}$, $h = 1, \ldots, 2^{n^2}$, be positive definite.

---

[1] For the sake of readibility, but with a slight abuse of notation, we use the symbol "p" to denote both a function and a generic element of $\mathbf{R}^\nu$; the context should make any confusion unlikely.

[2] By "on $\wp$" we mean "for all $p \in \wp$."

[3] Notice that the maximum and the minimum always exist in view of the hypotheses.

This result simplifies the task of checking condition (5), reducing it to a test of positivity for a finite number of matrices. As already said, however, the elements of $A(\cdot)$ do not vary independently of each other as is implicitly assumed in the formulation of the previous result. This can add some conservatism to this sufficient condition. As it will be illustrated in what follows, a smaller number of positivity tests is required if we take into account the dependence of the elements a_{ij} on the vector of parameters. A smaller number of positivity tests should bring, in general, less conservative results.

In order to see how, starting from the set of parameter variations, it is possible to select a finite number of matrices whose positive definiteness will guarantee the positive definiteness of $Q(p)$ on $\wp$, let us first introduce the following result [3].

Fact 2 *Consider the* symmetric *matrix function*

$$N(\rho) \triangleq \sum_{(i_1,\ldots,i_\mu)\in B^\mu} N_{i_1,\ldots,i_\mu} \rho_1^{i_1} \cdots \rho_\mu^{i_\mu} \tag{7}$$

with $N_{i_1,i_2,\ldots,i_\mu}$ constant symmetric matrices, and $\rho \triangleq [\rho_1,\ldots,\rho_\mu]^T$ a vector ranging in a hyperrectangle

$$\mathcal{R} \triangleq \{\rho \in \mathbf{R}^\mu | 0 \le \underline{\rho} \le \rho \le \overline{\rho}\}. \tag{8}$$

Matrix $N(\rho)$ is positive definite on $\mathcal{R}$ iff *it is positive definite in correspondence of the vertices $\rho^{(i)}$, $i = 1,\ldots,2^\mu$, of $\mathcal{R}$, i.e.*

$$N(\rho^{(i)}) > 0, \quad i = 1,\ldots,2^\mu. \tag{9}$$

This result is interesting because it refers to the vertices of a hyperrectangle in the parameter space while the result stated in Fact 1 refers to the vertices of a hyperrectangle in the space of the elements of the matrix. Unfortunately it does not directly apply to our problem since matrix $Q(p)$ in (5) does not possess the form (7). Indeed, recalling (3), it is possible to find symmetric matrices $\tilde{Q}_{i_1,\ldots,i_\nu}$, with $i_1,\ldots,i_\nu \in J \triangleq \{0,1,2\}$, so as to rewrite (5) as

$$Q(p) = -\sum_{(i_1,\ldots,i_\nu)\in J^\nu} \tilde{Q}_{i_1,\ldots,i_\nu} p_1^{i_1} \cdots p_\nu^{i_\nu}, \tag{10}$$

from which it is clear that, in general, $Q(\cdot)$ is not multilinear in the components of p.

3 Main Result

The use of Fact 2 for the solution of our problem is made possible by "immersing" the range of $Q(\cdot)$ in a set which, in turn, is the range of a multilinear function with a hyperrectangle as domain. To show this, for each $j = 1,\ldots,\nu$, and for each $p_j \in [\underline{p}_j, \overline{p}_j]$, let us introduce a $\pi_j(p_j)$ such that

$$p_j^2 = \pi_j(p_j)(p_j - \underline{p}_j) + \underline{p}_j^2. \tag{11}$$

The value of π_j depends on the value of p_j, and hence π_j is, in principle, unknown. On the other hand, the convexity of function p_j^2 and the *a priori* knowledge that p_j

varies in the interval $[\underline{p}_j, \overline{p}_j]$ enable us to conclude that π_j can only range in an interval $[\underline{\pi}_j, \overline{\pi}_j]$ with

$$\underline{\pi}_j \triangleq 2\underline{p}_j, \tag{12a}$$

$$\overline{\pi}_j \triangleq \frac{\overline{p}_j^2 - \underline{p}_j^2}{\overline{p}_j - \underline{p}_j}. \tag{12b}$$

We can then consider the substitution

$$p_j^2 \to \pi_j(p_j - \underline{p}_j) + \underline{p}_j^2; \tag{13}$$

define the *augmented* vector of uncertain parameters $\hat{p}$:[4]

$$\hat{p} \triangleq [p^T, \pi_1, \ldots, \pi_\eta]^T \in \mathbf{R}^{\hat{\nu}}, \qquad \hat{\nu} \leq 2\nu, \tag{14}$$

and, correspondingly, the hyperrectangle $\hat{\wp}$ in the augmented parameter space $\mathbf{R}^{\hat{\nu}}$

$$\hat{\wp} \triangleq \{\hat{p} \in \mathbf{R}^{\hat{\nu}} \mid \underline{\hat{p}} \leq \hat{p} \leq \overline{\hat{p}}\}, \tag{15}$$

where $\underline{\hat{p}} \triangleq [\underline{p}^T, \underline{\pi}_1, \ldots, \underline{\pi}_\eta]^T$ and $\overline{\hat{p}} \triangleq [\overline{p}^T, \overline{\pi}_1, \ldots, \overline{\pi}_\eta]^T$.

Now, let

$$\hat{Q}(\hat{p}) = - \sum_{(i_1,\ldots,i_{\hat{\nu}}) \in \mathcal{B}^{\hat{\nu}}} \hat{Q}_{i_1,\ldots,i_{\hat{\nu}}} \hat{p}_1^{i_1} \cdots \hat{p}_{\hat{\nu}}^{i_{\hat{\nu}}} \tag{16}$$

(where $\hat{p}_i$, $i = 1, \ldots, \hat{\nu}$, are the components of vector $\hat{p}$ introduced in (14)) be the multilinear matrix function obtained by applying (13) and (14) to Q.

We are in position to state the following result.

Theorem 1 *Assume A of the form (3) and consider the matrix functions Q and $\hat{Q}$. Then $Q(p)$ is positive definite on $\wp$* if *matrix $\hat{Q}(\hat{p})$ is positive definite in the $2^{\hat{\nu}}$ vertices of $\hat{\wp}$.*

Proof In view of (11), it is not difficult to see that

$$\{\hat{Q}(\hat{p}) \mid \hat{p} \in \hat{\wp}\} \supseteq \{Q(p) \mid p \in \wp\}. \tag{17}$$

Then, the positive definiteness of $\hat{Q}(\hat{p})$ on $\hat{\wp}$, which follows by use of Fact 2, implies the positive definiteness of $Q(p)$ on $\wp$. ∎

Notice that the application of this theorem is convenient with respect to the application of the result contained in Fact 1 when $\nu \leq n^2/2$ which, most of the time, applies in real world situations.

It can be also readily understood that the class of matrix functions $A(\cdot)$ for which results analogous to Theorem 1 hold, can be widened by assuming A to be a multivariate polynomial matrix in the component of p, i.e.

$$A(p) = \sum_{(i_1,\ldots,i_\nu) \in \mathcal{K}^\nu} A_{i_1,\ldots,i_\nu} p_1^{i_1} \cdots p_\nu^{i_\nu}, \tag{18}$$

---

[4] Note that, in general, $\eta \leq \nu$ since not necessarily all the components of p appear squared in Q.

with $K \triangleq \{0,1,\ldots,m\}$. Indeed, in this case we can generalize equations (11), (12) and (13) as follows: for each $j = 1,\ldots,\nu$, for each $p_j \in [\underline{p}_j, \overline{p}_j]$, and for each $i_j > 1$, introduce a $\pi_{ji_j}(p_j)$ such that

$$p_j^{i_j} = \pi_{ji_j}(p_j)(p_j - \underline{p}_j) + \underline{p}_j^{i_j}, \tag{19a}$$

$$\underline{\pi}_{ji_j} \triangleq i_j \underline{p}_j^{i_j-1}, \tag{19b}$$

$$\overline{\pi}_{ji_j} \triangleq \frac{\overline{p}_j^{i_j} - \underline{p}_j^{i_j}}{\overline{p}_j - \underline{p}_j}. \tag{19c}$$

Of course, the dimension of the enlarged vector $\hat{p}$ must be suitably increased. Details on this can be found in [5].

4 Conclusions

In [3] a result is presented which ensures Hurwitz-stability of a class of matrices having elements which are multilinear functions of a set of bounded parameters. This condition employs a finite number of tests. In this paper we have shown that that result can be used to check Schur-stability of the same class of matrices provided a suitable redefinition of the space of parameters and, consequently, a redefinition of the hyper-rectangle of parameter variation is made. A similar procedure can be used to deal with the general case of multivariate polynomial matrices.

Acknowledgements: The authors wish to thank prof. G. Celentano for his helpful suggestions and comments.

References

[1] M. Mansour, “Robust Stability of Interval Matrices,” *Proc. 28th CDC*, Tampa, Florida, 1989.

[2] H. Kiendl, “Totale Stabilität von Linearen Regelungssystemen bei ungenau bekannten Parametern der Regelstrecke,” *Automatisierungstechnik*, **33**, 1985.

[3] H. P. Horisberger and P. R. Bélanger, “Regulators for Linear Time Invariant Plants with Uncertain Parameters,” *IEEE Trans. Automat. Contr.*, **AC-21**, 1976.

[4] G. Ambrosino, G. Celentano, F. Garofalo, and L. Glielmo, “A Stability Robustness Test via Quadratic Lyapunov Forms,” *Proc. 28th CDC*, Tampa, Florida, 1989.

[5] F. Garofalo, G. Celentano, and L. Glielmo, “Stability Robustness of Interval Matrices via Quadratic Lyapunov Forms,” *submitted for publication*, 1990.

[6] R. E. Kalman and J. E. Bertram, “Control System Analysis and Design via the Second Method of Lyapunov (Discrete-Time Systems),” *ASME J. Basic Eng.*, 1960.

[7] T. Mori and H. Kokame, “Convergence Property of Interval Matrices and Interval Polynomials,” *Int. J. Control*, **45**, 1987.

Minimum Weight Quadrilateralization in $O(n^3 \log n)$ Time

HAROLD E. CONN
Supercomputing Research Center, 17100 Science Drive, Bowie, MD 20715
JOSEPH O'ROURKE
Smith College, Department of Computer Science, Northampton, MA 01063

ABSTRACT

We present an algorithm to find a minimum weight quadrilateralization of a convex polygon in $O(n^3)$ time, the same complexity as the best known algorithm for minimum weight triangulation of a convex polygon; and an algorithm for minimum weight quadrilateralization of an arbitrary simply-connected polygon in $O(n^3 \log n)$ time, improving on the $O(n^4)$ time complexity of an algorithm due to Keil and Sack [KS].

1. Introduction

The problem addressed in this paper is that of partitioning a simply-connected polygon into quadrilaterals in such a way that the sum of the edge lengths of the quadrilaterals is minimized. This type of partition is called a *minimum ink*, or *minimum weight* partition. We will be interested in partitioning into *convex* quadrilaterals, and the term *quadrilateralization* of a polygon will be used throughout this paper to refer to a partition into convex quadrilaterals.

In general, it is not always possible to partition a simply-connected polygon into convex quadrilaterals. However, Kahn *et al.* [KKK] proved that every *orthogonal* polygon does admit such a partition. In [ST] a linear time algorithm is presented for quadrilateralizing an orthogonal star-shaped polygon. An $O(n \log n)$ algorithm for quadrilateralizing an arbitrary simply-connected orthogonal polygon is presented in [S]. The first analysis of the minimum weight quadrilateralization problem is given in [KS], where an $O(n^4)$ dynamic programming algorithm for finding a minimum weight quadrilateralization of a simply-connected orthogonal polygon is developed. A similar algorithm is presented in [L], along with several other results regarding quadrilateralizations, including a proof that $O(n \log n)$ is a lower bound for quadrilateralizing a multiply-connected orthogonal polygon, and a proof that the problem of deciding whether a multiply-connected polygon can be quadrilateralized is NP-complete.

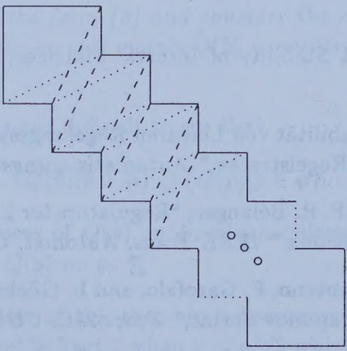


Figure 1. Exponential number of partitions.

As pointed out in [KS], the number of possible quadrilateralizations for a given polygon can be exponential in the number of vertices, as illustrated by the polygon displayed in Figure 1. Thus a brute force approach to solving the problem would, in the worst case, require examining an exponential number of possible partitions.

In the quadrilateralizations considered here, Steiner points (points that are not vertices of the original polygon) are not allowed. Thus vertices of quadrilaterals must coincide with vertices of the polygon. It is permissible for vertices to be collinear, so that the partition illustrated in Figure 2(a) is a valid quadrilateralization. On the other hand, we do not allow as a valid quadrilateralization a partition that contains a degenerate quadrilateral as illustrated in Figure 2(b). The chord d (shown in the figure as a curved arc) between the two collinear convex vertices forms a degenerate quadrilateral with the polygon edges a and c and the chord b. We exclude such a partition from consideration as a valid quadrilateralization.

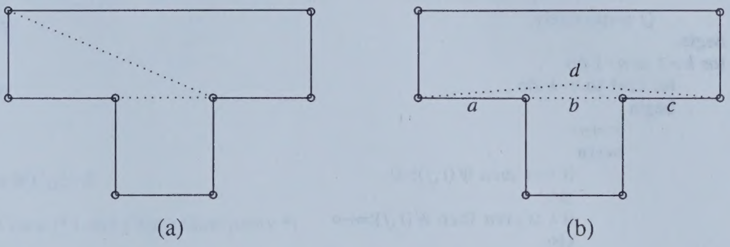


Figure 2. (a) Valid, (b) invalid quadrilateralizations.

In Section 3 we present an $O(n^3)$ dynamic programming algorithm for finding a minimum weight quadrilateralization of a convex polygon. This algorithm has the same complexity as the best known algorithm for minimum weight triangulation of a convex polygon.

In Section 4 we generalize the algorithm so that it is capable of finding a minimum weight quadrilateralization of a simply-connected polygon that is not convex. The time complexity of the general algorithm is $O(n^3 \log n)$ as compared to the $O(n^4)$ algorithm presented in [KS]. This improvement in time complexity is achieved at a cost of increasing the space complexity from $O(n^2)$ to $O(n^3)$.

2. Preliminaries

The algorithms in this paper operate on polygons that are specified as a list of vertices, where the vertices are numbered from 1 to n consecutively, in the order imposed by a clockwise traversal of the perimeter of the polygon. We are interested in partitioning such polygons into quadrilaterals.

The following lemma regarding quadrilateralizations is from [OR].

Lemma 1: [OR] If an n-vertex polygonal region with h holes can be decomposed into q quadrilaterals then $q=\frac{1}{2}(n+2h-2)$. In particular, n is even.

We are interested mainly in the fact that n must be even. Thus a polygon with an odd number of vertices does not admit a quadrilateralization. This is the key fact that we rely on in our algorithm for quadrilateralization of a convex polygon.

In [KS] Keil and Sack developed an algorithm for finding a minimum weight quadrilateralization of an orthogonal polygon in $O(n^4)$ time. Their algorithm can also be applied to non-orthogonal polygons. In case the input polygon does not admit any partition into convex quadrilaterals, the algorithm returns a value of $+\infty$. We now present their algorithm.

In the following algorithm $W(i,j)$ is the weight of the minimum weight quadrilateralization of the subpolygon P_{ij} cut off from p_i to p_j. (P_{ij} is the polygon formed by the perimeter of the original polygon moving clockwise from p_i to p_j inclusive, and with an edge drawn between p_i and p_j.)

As a preprocessing step in the following algorithm (and in all subsequent algorithms in this paper), we calculate the values d_{ij}, each of which is the Euclidean distance from p_i to p_j if the points are mutually visible, and is $+\infty$ otherwise. Note that the values d_{ij} can be calculated in $O(n^2)$ preprocessing time.

Algorithm 1 calculates $O(n^2)$ values $W(i,j)$ and each of these requires the consideration of $O(n^2)$ pairs (l,m) that could form a convex quadrilateral p_i, p_l, p_m, p_j. This results in a total time complexity of $O(n^4)$ and a space complexity of $O(n^2)$.

Algorithm 1. Minimum Weight Quadrilateralization - [KS]

```
Input: A polygon $P$
Output: The weight of the minimum weight quadrilateralization and
        the quadrilateralization itself are stored in $W(1,n)$ and
        $Q$ respectively.
begin
for $k$:=1 to $n-1$ do
    for $i$:=1 to $n-k$ do
    begin
        $j:=i+k$
        begin
            if $k=1$ then $W(i,j):=0$
            else
            if $k$ is even then $W(i,j):=+\infty$
            else
            begin
                $W(i,j):=d_{ij}+min(W(i,l)+W(l,m)+W(m,j))$
                (* where the minimum is taken over all $l$
                  and $m$ such that
                  (a) $i<l<m<j$, and
                  (b) $p_i,p_l,p_m,p_j$
                      form a convex quadrilateral.
                *)
                $Q(i,j):=(l,m)$
                (* indices of vertices achieving the minimum
                  are stored in Q *)
            end
        end
    end
end
```

3. Quadrilateralization of a Convex Polygon

Algorithm 2 finds a minimum weight partition into convex quadrilaterals of an arbitrary convex polygon. It is based on Algorithm 1 with the following modifications.

At each step of the innermost loop of Algorithm 1, an attempt is made to choose values of l and m so that the points p_i, p_l, p_m, and p_j form a convex quadrilateral. The value of the variable k is $j-i$, corresponding to the chord from p_i to p_j. When k is even, $W(i,j)$ is set to infinity. This is because the corresponding chord cuts off a subpolygon with an odd number of vertices, and such a polygon does not admit a partition into convex quadrilaterals.

The variable k is even at every other diagonal of the table $W(i,j)$ as it is filled in by the dynamic programming algorithm. At each step in each such diagonal, the algorithm does constant work, *viz.* $W(i,j):=+\infty$. Then at each step of every diagonal where k is odd, the algorithm does worst case $O(n^2)$ work: the minimization over $O(n^2)$ pairs of candidates for l and m. This results in an overall worst case time complexity of $O(n^4)$ for Algorithm 1.

We now show that by doing $O(n)$ work in the even diagonals of the table, the amount of work required in the odd diagonals can be reduced to $O(n)$, resulting in an overall time complexity of $O(n^3)$ for finding a quadrilateralization. When the variable k is even, the algorithm calculates a minimum weight partition of the subpolygon P_{ij} into convex quadrilaterals with the exception that a single triangle with vertices at p_i and p_j must be included in the partition. This calculation requires linear time.

Each of the two minimization steps in the innermost loop requires linear work. The loop is executed $O(n^2)$ times, so the overall complexity of the algorithm is $O(n^3)$.

The correctness of the algorithm follows from the fact that at each stage a minimum weight quadrilateralization of subpolygon P_{ij} is found by minimizing the sum $W(i,m)+W(m,j)$, where m has the same parity as i, and m and j have opposite parity. The quantity $W(m,j)$ represents a minimum weight quadrilateralization of subpolygon P_{mj} and the quantity $W(i,m)$ represents a minimum weight partitioning of subpolygon P_{im} into a single triangle and the balance into quadrilaterals as previously explained. Each of the two quantities will have

Algorithm 2. Minimum Weight Quadrilateralization of a Convex Polygon

Input: A convex polygon P

Output: The weight of the minimum weight quadrilateralization and the quadrilateralization itself are stored in $W(1,n)$ and Q respectively.

```
begin
for $k$:=1 to $n-1$ do
    for $i$:=1 to $n-k$ do
    begin
        $j:=i+k$
        begin
            if $k=1$ then $W(i,j):=0$
            else
            if $k$ is even then (* $i$ and $j$ have same parity *)
            begin
                $W(i,j):=min(W(i,l)+W(l,j))$
                (* where the minimum is taken over all $l$
                  such that
                  (a) $i<l<j$
                  (b) $l$ has parity opposite to $i$ and $j$
                *)
                $L(i,j):=l$
                (* save index of vertex achieving minimum *)
            end
            else (* $k$ odd - $i$ and $j$ have opposite parity *)
            begin
                $W(i,j):=d_{ij}+min(W(i,m)+W(m,j))$
                (* where the minimum is taken over all $m$
                  such that
                  (a) $i<m<j$, and
                  (b) $m$ has the same parity as $i$
                *)
                $Q(i,j):=(L(i,m),m)$
                (* indices of vertices achieving the minimum
                  are stored in Q
                *)
            end
        end
    end
end
```

been calculated at a previous stage of the algorithm. The minimization over the sum of the two quantities must result in a minimum weight quadrilateralization of subpolygon P_{ij}. The resulting quadrilateralization may include only convex quadrilaterals since the input polygon is convex.

Theorem 1: Algorithm 2 finds a minimum weight partition of a convex polygon into convex quadrilaterals in $O(n^3)$ time and $O(n^2)$ space.

4. The General Algorithm

The structure of the general algorithm for finding a minimum weight partition into convex quadrilaterals is similar to that of Algorithm 2. However, in order to ensure that all the quadrilaterals in the partition are convex, the algorithm must do some additional work. We show that by doing $O(n \log n)$ (amortized) work in the even diagonals of the table $W(i,j)$, the amount of work required in the odd diagonals can be reduced to $O(n)$, resulting in an overall time complexity of $O(n^3 \log n)$. The method to be described requires the use of balanced binary tree data structures. For every pair of vertex indices (i,j) where $k=j-i$ is even, we maintain a balanced binary tree T_{ij}.

The entries in T_{ij} are values of the form $W(i,q)+W(q,j)$ where q is a vertex index in the range $i<q<j$, and where i and j have the same parity and q has opposite parity. These values represent the sum of the weights of the minimum weight quadrilateralizations of the subpolygons P_{iq} and P_{qj}. The value of the vertex

index q is also stored in the tree. The minimum value stored in T_{ij} will be used at a later step of the algorithm as one possible choice in the calculation of the minimum weight quadrilateralization of a larger subpolygon P_{is}. See Figure 3.

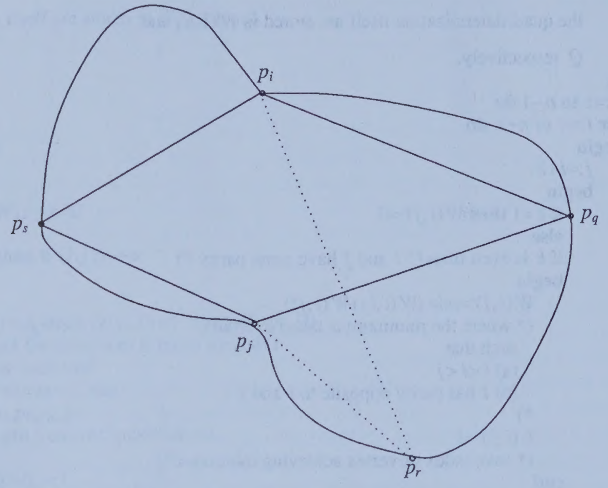


Figure 3. Calculation of T_{ij}

As can be seen from Figure 3, the value $W(i,r)+W(r,j)$ is not of interest when calculating the minimum weight quadrilateralization of P_{is} because p_r is not visible from p_s, and thus the quadrilateral $p_i p_r p_j p_s$ is not convex. Because of this, we must ensure that at the time of the calculation of the minimum weight quadrilateralization of the subpolygon P_{is}, the minimum value stored in T_{ij} corresponds to a point that is visible from the point p_s. If it does not, then the minimum value must be deleted from the tree. This means that the data structure T_{ij} must support dynamic insertions and deletions. The notation $T_{ij}(x)$ is used to indicate that the values being inserted into the tree T_{ij} correspond to vertices that are visible from the vertex p_x.

The basic operations to be used in conjunction with the balanced tree structures are INSERT, DELETE, and MINIMUM. They are used as follows. The operation INSERT(T_{im},$(W(i,l)+W(l,m))$) stores in balanced tree T_{im} both the value $(W(i,l)+W(l,m))$ and the corresponding vertex index l. The operation DELETE(T_{im},l) deletes from tree T_{im} the information for p_l. The operation (l,opt):=MINIMUM(T_{im}) assigns to variable *opt* the minimum of any of the values stored in T_{im}, and assigns to variable l the value of the vertex index at which the minimum is achieved.

A series of $O(n)$ of these three types of instructions can be performed on a balanced binary tree in $O(n \log n)$ time and $O(n)$ space. See, for example, [AHU].

We now present the algorithm for finding a minimum weight quadrilateralization of a general simply-connected polygon. Because the algorithm is somewhat complicated, we first present a ''pseudo-code'' version where the operations involving the balanced tree structures are described only from a high level point of view. The details of the tree operations will be filled in later.

Note that in the high level description of the algorithm the process of ''initializing'' a tree T_{ij} is shown as occurring when the program variable k is even. Conceptually, it is convenient to think of the algorithm operating this way, but in the detailed description of the algorithm no distinction is made between initializing a tree and updating a tree. Furthermore, all operations on all trees are performed when the program variable k is odd, but by an amortization process they are ''charged'' to even values of the program variable k.

In order to guarantee that at most $O(n \log n)$ time is needed for maintaining each of the balanced trees, it is necessary to show that each tree requires at most $O(n)$ MINIMUM, INSERT, and DELETE operations. It is clear that at most $O(n)$ MINIMUM operations are required for each tree. It must be shown that insertions and deletions have a monotonicity property that will prevent a given vertex from being repeatedly considered for

Algorithm 3. Minimum Weight Quadrilateralization - High Level

Input: A polygon P

Output: The weight of the minimum weight quadrilateralization and the quadrilateralization itself are stored in $W(1,n)$ and Q respectively.

```
begin
for $k$:=1 to $n-1$ do
   for $i$:=1 to $n-k$ do
   begin
      $j:=i+k$
      begin
         if $k=1$ then $W(i,j):=0$
         else
         begin
            if $k$ is even then
            begin
               $W(i,j):=+\infty$
               INITIALIZE $T_{ij}$
               (* calculate $T_{ij}(j+1)$ *)
               (* include values corresponding to vertices that
                  are visible from $p_{j+1}$     *)
            end
            else (* $k$ is odd *)
            begin
               for $m:=i+2$ to $j-1$ by 2 do
               begin (* update $T_{im}$ for vertex $j$ *)
               (* add vertices that are newly visible to $p_j$ *)
               (* delete vertices no longer visible to $p_j$ *)
               $(L(m),OPT(m))$:=MINIMUM($T_{im}$)
               (* assign to $OPT(m)$ the minimum value in $T_{im}$ *)
               (* $L(m)$ contains the index of the vertex that achieves the min *)
               (* charge all operations in this begin block to the pair $(i,m)$ *)
               end
            $W(i,j):=d_{ij}+W(m',j)+min(OPT(m))$
            (* where the minimum is taken over all values of $m$ as in
              the preceding for loop. $m'$ is the value of
              $m$ at which the minimum is achieved.
            *)
            $Q(i,j):=(L(m'),m')$
            (* indices achieving the minimum are stored in Q *)
            end
         end
      end
   end
end
```

insertion into a given tree, and from being repeatedly inserted into and deleted from the same tree.

Now we prove several lemmas which characterize this monotonicity property and thus limit the number of times a vertex needs to be considered for insertion into and deletion from a given tree.

First note that, given four vertices of P, say p_i, p_l, p_m, and p_j with $i<l<m<j$, if p_l is visible to p_i and p_m, and if p_j is visible to p_i and p_m, then the four vertices may or may not form a convex quadrilateral. If they fail to form a convex quadrilateral, then the visibility constraints imply that they do so because the quadrilateral $p_i p_l p_m p_j$ has a reflex angle at either p_i or at p_m. In case the reflex angle occurs at p_i, we say that visibility between p_j and p_l is *blocked* by p_i. Similarly for p_m.

We summarize these observations in the following lemma:

Lemma 2: Given any polygon P and four of its vertices p_i, p_l, p_m, p_j with $i<l<m<j$, if p_l is visible to p_i and p_m, and if p_j is visible to p_i and p_m, then either p_j and p_l are mutually visible or visibility between them is blocked by vertex p_i or by vertex p_m.

Proof: If p_j and p_l are not mutually visible, and if visibility between them were blocked by some vertex p_x distinct from p_i or p_m, then p_x must also block visibility between at least one of the pairs $p_i p_j$, $p_j p_m$, $p_i p_l$, or $p_l p_m$. This would contradict the hypothesis of the lemma. □

Lemma 2 says, in effect, that the visibility relationship between p_j and p_l falls into one of three cases:

(a) p_j and p_l are mutually visible, or

(b) visibility between p_j and p_l is blocked by p_m, or

(c) visibility between p_j and p_l is blocked by p_i.

The following lemma provides further information regarding case (b).

Lemma 3: Given a polygon P and four of its vertices p_i, p_l, p_m, p_j with $i<l<m<j$, and such that p_l is visible to p_i and p_m, and p_j is visible to p_i and p_m. Then if visibility between p_l and p_j is blocked by p_m, then no other vertex p_x with $l<x<m$ is visible from p_j.

Proof: Assume that the polygon is oriented as in Figure 4, and consider the line l_{jm} through p_j and p_m. Let p_x be a vertex with $l<x<m$. If either of the following two conditions is violated, then p_x cannot be visible to p_j: (i) p_x must be above or coincident with the line l_{jm}, and (ii) the line segment $p_j p_x$ must be internal to the polygon P.

Clearly, in the case illustrated by Figure 4(a) where p_x is below line l_{jm} in violation of condition (i), p_x is not visible from p_j.

Although it is possible for a point p_x with $l<x<m$ to be situated above line l_{jm}, in any such case the line segment $p_x p_j$ cannot be internal to P, as illustrated in Figure 4(b).

Finally, p_x can be placed so that it is not in violation of either of the conditions, as illustrated in Figure 4(c). But in any such situation, the visibility of p_l from p_m will be blocked by p_x in violation of the hypothesis of the lemma. □

The next lemma further characterizes case (c) above:

Lemma 4: Given a polygon P and four of its vertices p_i, p_l, p_m, p_j with $i<l<m<j$, and such that p_l is visible to p_i and p_m, and p_j is visible to p_i and p_m. Then if visibility between p_l and p_j is blocked by p_i, then no other vertex p_y with $n \geq y > j$ is visible from p_l. (Recall that the vertices of P are numbered from 1 to n.)

Proof: Analogous to the proof of the preceding lemma. □

In fact, none of the vertices "between" p_j and p_i can be visible to p_l. But at the stage of the algorithm(s) where this lemma will be applied, the vertices $p_1 \cdots p_{i-1}$ are no longer of interest. The situation is illustrated in Figure 5.

4.1. Insertions into T_{im}

With the three preceding lemmas, we now have enough machinery to analyze insertions into the balanced binary trees.

Suppose we are updating tree T_{im} with respect to vertex p_j. We consider, as possible entries into the tree, vertices having indices between i and m that are visible to both p_i and p_m. (A vertex that is not visible to both p_i and p_m could not possibly serve as vertex p_l in a convex quadrilateral $p_i p_l p_m p_j$.) So the potential range of indices for a candidate vertex p_l is $i<l<m$.

As program variable l ranges from $i+1$ upward, there are the following cases:

(1) p_l is not visible to both p_i and p_m - then skip to next possible value of l, $l:=l+2$.

(2) p_l is visible to both p_i and p_m.

Subcase (a) p_j and p_l are mutually visible - then enter the value corresponding to p_l into tree T_{im}, and consider next value of l, $l:=l+2$.

Subcase (b) visibility between p_j and p_l is blocked by p_i. Then by Lemma 4, vertex p_l will not be visible to subsequent vertices having indices greater than j. Thus p_l can be skipped over, and will never again need to be considered for entry into tree T_{im}. Set $l:=l+2$.

Subcase (c) visibility between p_j and p_l is blocked by p_m. Then by Lemma 3, there is no other vertex between p_l and p_m that is visible to vertex p_j. So updating of tree T_{im} with respect to vertex p_j is completed. Save the current value of program variable l so that when we begin updating tree T_{im} with respect to vertex p_{j+2}, we don't have to start over from vertex p_{i+1} again.

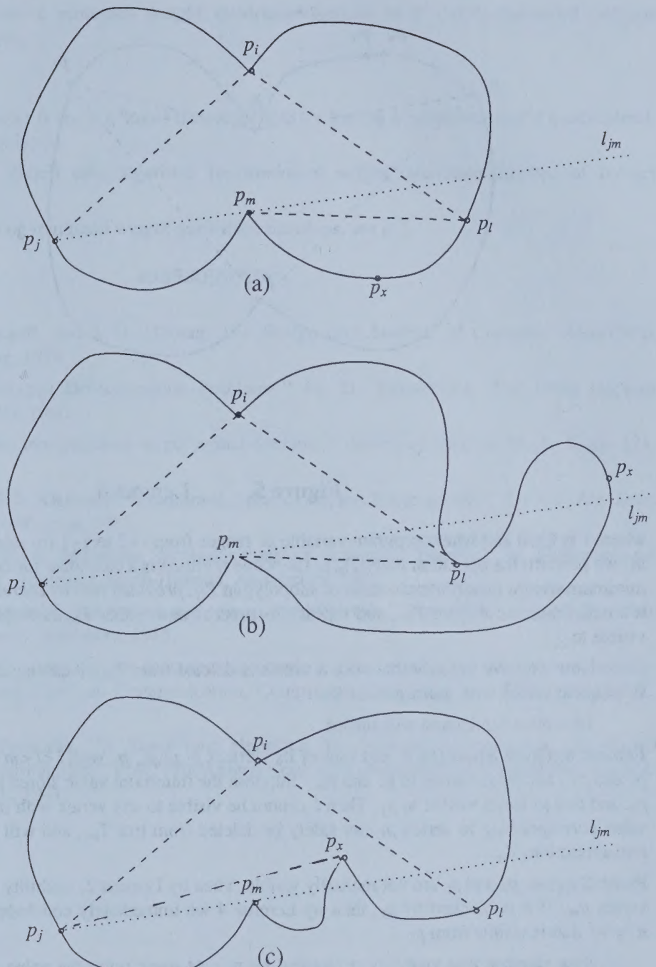


Figure 4. Proof of Lemma 3.

4.2. Deletions from T_{im}

We now prove another lemma, similar to Lemmas 3 and 4, that will be used in analyzing deletions from a tree T_{im}.

Lemma 5: Given a polygon P and four of its vertices p_i, p_l, p_m, p_j with $i < l < m < j$, and such that p_l is visible to p_i and p_m, and p_j is visible to p_i and p_m. Then if visibility between p_l and p_j is blocked by p_m, then no other vertex p_z with $m < z < j$ is visible from p_l.

Proof: The proof is entirely analogous to that of Lemma 3. □

In fact this is really just a restatement of Lemma 3 with the orientation of the polygon changed and vertices relabeled. But the following argument is made clearer by having a distinct lemma that can be cited without needing to consider rotating the polygon P and relabeling vertices.

Deletions from a tree T_{im} occur "on demand." That is, when we are attempting to find a minimum weight quadrilateralization of subpolygon P_{ij}, we look at all possible minimum values stored in the trees T_{im},

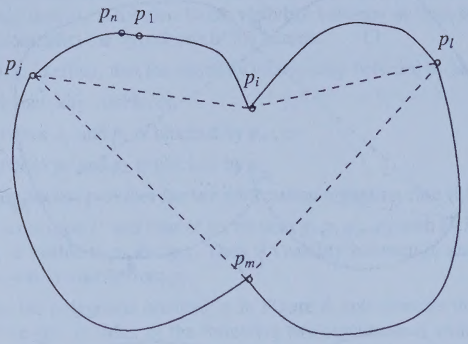


Figure 5. Lemma 4

where i is fixed and where program variable m ranges from $i+2$ to $j-1$ (in steps of 2.) So, for a given value of m, we perform the operation MIN(T_{im}). The value returned is a candidate for consideration in the search for the minimum weight quadrilateralization of subpolygon P_{ij}, *provided that its corresponding vertex is visible to j.* If it's not, we delete it from T_{im}, and repeat this process until either T_{im} is empty or we get a "winner" that is visible to p_j.

Now we have to show that once a vertex is deleted from T_{im}, it can never serve as a "winner" for some subsequent vertex with index greater than j.

The following lemma will suffice.

Lemma 6: Given a polygon P and four of its vertices p_i, p_l, p_m, p_j with $i<l<m<j$, and such that p_l is visible to p_i and p_m, and p_j is visible to p_i and p_m. Suppose the minimum value stored in tree T_{im} corresponds to vertex p_l, and that p_l is not visible to p_j. Then l cannot be visible to any vertex with index greater than j, and thus the value corresponding to vertex p_l can safely be deleted from tree T_{im}, and will never need to be considered for reinsertion into T_{im}.

Proof: Suppose p_j and p_l are not mutually visible. Then by Lemma 2, visibility is blocked by either vertex p_i or vertex p_m. If it is blocked by p_i, then by Lemma 4 we immediately conclude that there is no vertex p_y with $n \geq y > j$ that is visible from p_l.

Now suppose that visibility is blocked by p_m. At some point the value corresponding to vertex p_l was inserted into the tree T_{im}. This event must have been "triggered" by some vertex p_z with $j>z>m$. But this contradicts Lemma 5. ☐

4.3. Detailed Description of Algorithm 3

For a detailed pseudo-code implementation of Algorithm 3, see [C].

Each tree T_{ij} undergoes a series of at most $O(n)$ INSERT, DELETE, and MINIMUM operations. (That the number of INSERT and DELETE operations is linear is guaranteed by Lemma 2.) Thus maintaining each tree requires $O(n \log n)$ time. For tree T_{ij}, these operations are charged to the pair (i,j) where i and j have the same parity, corresponding to even values of the variable k. There are $O(n^2)$ different trees, and thus maintaining all of the trees requires $O(n^3 \log n)$ time. When the variable k is odd, a linear amount of work is needed for the minimization of the array OPT which contains $O(n)$ values. Thus the time complexity of the algorithm is dominated by the maintenance of the trees and is $O(n^3 \log n)$.

This improvement in time complexity is achieved at the cost of an increase in the space required. Algorithm 1 required only $O(n^2)$ space for maintaining the array $W(i,j)$. The new algorithm in addition requires linear space for each of the $O(n^2)$ trees, and thus its space complexity is $O(n^3)$.

Theorem 2: Algorithm 3 finds a minimum weight quadrilateralization of a simply-connected polygon $O(n^3 \log n)$ time and $O(n^3)$ space.

5. Conclusion

The main result of this paper is an $O(n^3 \log n)$ time algorithm for finding a minimum weight quadrilateralization of a simply-connected polygon.

We also developed an $O(n^3)$ time algorithm for minimum weight quadrilateralization of convex polygons.

For other results concerning minimum weight quadrilateralizations, see [C].

REFERENCES

[AHU] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, 1974.

[C] H. E. Conn, "Some Polygon Decomposition Problems," Ph. D. Dissertation, The Johns Hopkins University, Baltimore, MD, 1990.

[K] G. T. Klincsek, "Minimal triangulations of polygonal domains," *Annals of Discrete Math.*, 9, pp. 121-123, 1980.

[KKK] J. Kahn, M. Klawe, and D. Kleitman, "Traditional galleries require fewer guards," *SIAM J. Alg. Disc. Meth.*, vol. 4, 2, pp. 194-206, June, 1983.

[KS] J. M. Keil and J.-R. Sack, "Minimum decompositions of polygonal objects," in *Computational Geometry*, edited by G. T. Toussaint, North-Holland, Amsterdam, 1985.

[L] A. Lubiw, "Decomposing polygonal regions into convex quadrilaterals," Proc. First Annual Symposium on Computational Geometry, Baltimore, 1985.

[S] J.-R. Sack, "An $O(n \log n)$ algorithm for decomposing simple rectilinear polygons into convex quadrilaterals," Proc. 20th Annual Conf. on Communication, Control and Computing, pp. 64-75, Urbana, IL, 1982.

[ST] J.-R. Sack and G. T. Toussaint, "A linear time algorithm for decomposing rectilinear star-shaped polygons into convex quadrilaterals," Proc. 19th Annual Conf. on Communication, Control and Computing, pp. 21-30, Urbana, IL, 1981.

An Optimal Parallel Algorithm for the Rectilinear Voronoi Diagram

Sumanta Guha[1]
EECS Department, The University of Michigan, Ann Arbor, MI 48109-2122

ABSTRACT

We present the first optimal parallel algorithm to compute the Voronoi diagram in an L_p metric. The algorithm constructs the Voronoi diagram of n points on the plane under the rectilinear (L_1, Manhattan) metric on a CREW PRAM with $n/\log n$ processors in $O(\log^2 n)$ time. The processor-time product of $O(n \log n)$ is optimal and improves the previous best by a factor of $\log n$. Some new techniques are developed that should have applicability to many geometric problems under the rectilinear metric and, possibly, other L_p metrics.

1 Introduction

A centrally important construct in computational geometry is the *Voronoi diagram.* Given a set S of n points in the plane the Voronoi diagram of S is a plane graph with n faces and such that each face contains exactly one point of S in its interior and is the locus of those points in the plane which are as close to that point as to any other point of S. Using the Voronoi diagram several geometric proximity features of a point set may be determined. For example, as shown by Shamos [14], with the Voronoi diagram of a point set one can efficiently obtain the closest pair of points, the nearest neighbor of each point, the Euclidean minimum spanning tree of the points, the largest empty circle with center inside the convex hull of the points, and other features. The first optimal sequential algorithm to compute the Voronoi diagram was given by Shamos and Hoey [15] and takes $O(n \log n)$ time.

Owing to its importance several generalizations of the Voronoi diagram have been examined, see for example [13] and [16]. The generalization most relevant to this paper concerns Voronoi diagrams with distances measured by some L_p metric, $1 \leq p \leq \infty$, and not merely when $p = 2$ which gives the usual Euclidean metric. In fact, in many applications it is natural to measure distances with the L_1 metric. VLSI artwork is an example. Here current technology allows wires to be laid only in certain fixed directions, typically in two perpendicular directions. Other applications arise from computer graphics and robot motion planning. Sometimes also the simpler structure of a geometric problem under the L_1 metric is first studied to gain insight into the more difficult version with the Euclidean metric. Hwang [8] gives a Voronoi diagram algorithm for the L_1 metric, Lee and Wong [11] give an algorithm for the L_1 and L_∞ metrics, and Lee [10] gives an algorithm for the L_p metric. All these sequential algorithms run in optimal $O(n \log n)$ time and have applications similar to the Euclidean case. For example, the rectilinear Voronoi diagram leads to optimal computations of the rectilinear minimum spanning tree [8], nearest neighbors under the L_1 metric, and so on.

---

[1] Funded by NSF/DARPA grant CCR-9004727.

The problem we study in this paper is the computation of the Voronoi diagram on a parallel machine. The computation model we assume is the *CREW PRAM* – concurrent-read exclusive-write parallel random access machine. A CREW PRAM is a synchronous shared-memory multiprocessor machine such that any number of processors can simultaneously read one memory cell each in constant time, and any number of processors can simultaneously write one memory cell each in constant time except that no two processors may attempt to simultaneously write the same cell. The first parallel algorithm to compute the Euclidean Voronoi diagram was given by Chow [6] based on Brown's [5] reduction of the Voronoi diagram of a point set on the plane to the convex hull of a point set in 3-space and runs in $O(\log^3 n)$ time on a CREW PRAM with n processors. The time was improved to $O(\log^2 n)$ by Aggarwal *et al.* [1] using an algorithm based on divide-and-conquer. A similar but simpler algorithm was given by Jeong and Lee [9] to compute the Voronoi diagram on a $\sqrt{n} \times \sqrt{n}$ mesh-connected computer in optimal $O(\sqrt{n})$ time. Their algorithm, too, may be adapted, using Cole's [7] parallel sorting algorithm and the parallel planar point location method of Atallah, Cole, and Goodrich [2], to run in $O(\log^2 n)$ time on an n-processor CREW PRAM. Further, it may be modified to compute the L_p Voronoi diagram, $1 \leq p \leq \infty$, within the same processor-time bounds. However, the algorithms of Aggarwal *et al.* and Jeong and Lee having processor-time products of $O(n \log^2 n)$ are suboptimal by a factor $\log n$. We present an algorithm to compute the rectilinear Voronoi diagram in $O(\log^2 n)$ time on a CREW PRAM with $n/\log n$ processors which is optimal. Our method is inspired by that of Jeong and Lee but we introduce new ideas to avoid sorting and planar point location based on some very interesting interplay between the underlying geometry and basic parallel processing techniques.

For a cleaner presentation of our algorithm in Section 4 we first describe the underlying geometry in Section 2 and then collect needed results about parallel algorithms in Section 3. We conclude in Section 5 and present directions for future research.

2 Some Geometric Definitions and Facts

Given two points $q_1 = (x_1, y_1)$ and $q_2 = (x_2, y_2)$ on the plane the distance between them in the L_p metric, $1 \leq p < \infty$, is

$$d_p(q_1, q_2) = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{\frac{1}{p}}$$

and in the L_∞ metric is

$$d_\infty(q_1, q_2) = max(|x_1 - x_2|, |y_1 - y_2|).$$

Henceforth we shall only be concerned with the L_1 metric and denote the *distance* between q_1 and q_2 by

$$d(q_1, q_2) = |x_1 - x_2| + |y_1 - y_2|.$$

The *bisector* $B(q_1, q_2)$ of q_1 and q_2 is the locus of points equidistant from them, that is

$$B(q_1, q_2) = \{q \in E^2 : d(q, q_1) = d(q, q_2)\}.$$

When $|x_1 - x_2| > |y_1 - y_2|$ then $B(q_1, q_2)$ consists of two infinite vertical segments joined by an inclined segment making either 45° or 135° with the x-axis. This is called a *vertical bisector*. When $|x_1 - x_2| < |y_1 - y_2|$ then $B(q_1, q_2)$ is a *horizontal bisector* consisting of two infinite horizontal segments joined by an inclined segment. When $|x_1 - x_2| = |y_1 - y_2|$, to avoid bisectors with non-zero area, we specially define $B(q_1, q_2)$ to be a vertical bisector. See Figure 1.

Define

$$H(q_1, q_2) = \{q \in E^2 : d(q, q_1) \leq d(q, q_2)\},$$

a region homeomorphic to a closed half-plane with a boundary on $B(q_1, q_2)$.

Let $V(P)$ denote the Voronoi diagram of an n-point set $P = \{q_1, \ldots, q_n\}$. The *Voronoi polygon (face)* corresponding to a point $q_i \in P$ is $F_P(q_i) = \bigcap_{j \neq i} H(q_i, q_j)$, a (not necessarily bounded or convex) polygonal region whose edges are portions of bisectors. $V(P)$ is, in fact, a plane graph which is the union of the boundaries of the $F_P(q_i)$, $1 \leq i \leq n$. A point of $V(P)$ at which three of more bisectors meet is called a *Voronoi vertex.* When the portion of a horizontal bisector unbounded to the right belongs to $V(P)$ define (∞, y) to be a Voronoi vertex (the end at "infinity") of $V(P)$, too, where the horizontal segment going rightwards has ordinate y. Similarly define Voronoi vertices of the forms $(-\infty, y)$, (x, ∞), and $(x, -\infty)$. Denote the set of Voronoi vertices by $Ver(P)$. A portion of a bisector between two Voronoi vertices is called a *Voronoi edge.* A representation of $V(P)$ as a plane graph, see [13], is of size $O(n)$.

A simple but crucial geometric fact is the following:

Property 1 *Let E be any iso-oriented rectangle containing the points of P. The portions of the edges of $V(P)$ emerging from the left and right vertical sides of E are horizontal lines unbounded to the left and right, respectively. Similarly, the portions of the edges of $V(P)$ emerging from the top and bottom sides of E are vertical lines unbounded above and below, respectively.*

Proof Straightforward and omitted in this version but see Figure 2. □

Consider a partition of P into subsets L and R lying to the left and right, respectively, of a vertical line D. The *dividing chain C* of L and R is defined to be the set of edges in $V(P)$ each of which is shared by the Voronoi polygon of a point in L and the Voronoi polygon of a point in R. We shall also use C to denote the curve formed by the union of these edges. C may consist of several disjoint components. See Figure 3.

Property 2 *Each component of the dividing chain C of L and R is a polygonal line unbounded in both directions and monotone with respect to the y-axis. Linearly order each component so that it is monotone increasing with respect to the y-axis. Distinct components of C have disjoint projections on the y-axis so the linear orders of the components may be combined to impose a* linear order *on C by defining every point of one component to precede every point of another component that is above it.*

A point on C is as close to a point in L as it is to a point in R. Consider a point m not on C. Draw a straight line l from m in such a direction that it intersects C (choose any one of infinitely many such directions) and suppose that l first intersects C at some point in the component C'. Then m is closer to a point in L or to a point in R according as m is to the left or right of C'.

Proof Omitted in this version. □

The dividing chain thus "divides" the plane into two regions $H(L)$ and $H(R)$ (neither of which may be connected) which are the locus of points closer to L and to R, respectively. The next property sets the stage for divide-and-conquer:

Property 3 *$V(P)$ is the union of $V(L) \cap H(L)$, $V(R) \cap H(R)$, and C.*

Proof Follows directly from the definitions of $H(L)$, $H(R)$, and C. □

The following is a criterion to determine if a Voronoi edge is intersected by the dividing chain:

Property 4 *An edge $e \in V(L)$ intersects C if and only if one of the following conditions hold:*

1. *One endpoint of e is closer to L and the other is closer to R. In this case C intersects e in exactly one point.*
2. *Both endpoints of e are closer to R and the horizontal ray emanating leftwards from one of q_i or q_j, where e is a portion of $B(q_i, q_j)$, intersects e at at a point m (there can only be one such) closer to L. In this case C intersects e in exactly two points, one on either side of m. Call the segments of e on either side of m the two* semiedges *of e.*

A symmetric statement applies to edges $e \in V(R)$ with "L" switched with "R" and "leftwards" switched with "rightwards".

Proof Omitted in this version. □

Let M be the set consisting of all edges in $V(L)$ that intersect C once together with all semiedges (defined in Property 4) of edges in $V(L)$ that intersect C twice. Symmetrically define $N \subseteq V(R)$. Let $S = M \cup N$. Then by the previous property M, N, and S have consistent linear orders induced by their intersections with C (whose linear order is defined in Property 2). The next property helps decide the order of two given edges in S. But first, some more definitions. Suppose $e \in S$. By Property 4 one end vertex of e is closer to L and the other closer to R. Denote the vertex closer to L by $lv(e)$ and the other by $rv(e)$. Assume e is directed from $lv(e)$ to $rv(e)$. Extend the first and last straight segments of e infinitely. This extension partitions the plane into two closed unbounded regions, one to the left and one to the right of e. Any point or line is said to *lie left* or *lie right* of e according as it is contained wholly in the region to the left or right of e. Suppose e is a portion of the bisector $B(q_i, q_j)$ where q_i lies left of e and q_j lies right of e. Then denote q_i by $lp(e)$ and q_j by $rp(e)$. See Figure 3.

Property 5 *Suppose e_i and e_j are in S. Cases (1) and (2) determine the order of e_i and e_j.*

(1) *The projections of e_i and e_j on the y-axis are disjoint. Then $e_i > e_j$ or $e_i < e_j$ according as e_i is above or below e_j.*

(2) *The projections of e_i and e_j on the y-axis are not disjoint. Then the order of e_i and e_j is determined by cases (2.1) and (2.2).*

(2.1) *e_i intersects e_j at point m. Then the order of e_i and e_j is determined by cases (2.1.1) and (2.1.2).*

(2.1.1) *Both of e_i and e_j are either in M or in N. Then $e_i < e_j$ if e_i lies right of e_j; $e_j < e_i$, otherwise.*

(2.1.2) *One of e_i and e_j is in M and the other in N. Assume w.l.o.g. that $e_i \in M$ and $e_j \in N$. Then $e_i < e_j$ if $d(m, lp(e_i)) < d(m, lp(e_j))$ and $lv(e_j)$ lies right of e_i or if $d(m, lp(e_i)) > d(m, lp(e_j))$ and $lv(e_i)$ lies right of e_j; $e_j < e_i$, otherwise.*

(2.2) *e_i does not intersect e_j. Then one of e_i and e_j must lie to the left or right of the other. Then the order of e_i and e_j is determined by cases (2.2.1) and (2.2.2).*

(2.2.1) *One of e_i and e_j lies to the left of the other. Assume w.l.o.g. that e_j lies left of e_i. Then $e_j < e_i$ if $y(rv(e_j)) \leq y(lv(e_i)) \leq y(rv(e_i))$ or if $y(lv(e_j)) \leq y(rv(e_i)) \leq y(lv(e_i))$; $e_i < e_j$, otherwise.*

(2.2.2) *One of e_i and e_j lies to the right of the other. Assume w.l.o.g. that e_j lies right of e_i. Then $e_i < e_j$ if $y(lv(e_i)) \leq y(rv(e_i)) \leq y(lv(e_j))$ or if $y(rv(e_i)) \leq y(lv(e_i)) \leq y(rv(e_j))$; $e_j < e_i$, otherwise.*

Proof Omitted in this version but see [9]. □

The following property further illuminates the geometry underlying the ordering of C:

Property 6 *If C intersects a face f of $V(L)$ then the sequence of edges (and semiedges), ordered by C, at which it intersects the boundary of f is also in counterclockwise order around f and forms a subinterval of the ordered set M. Call this the M-*list *of f. Similarly, if C intersects a face g of $V(R)$ then the sequence of edges (and semiedges), ordered by C, at which it intersects the boundary of g is also in clockwise order around g and forms a subinterval of the ordered set N, the N-*list *of g.*

Proof Omitted in this version. □

The next property helps "locate" successive disjoint components of C:

Property 7 *Suppose C_1 and C_2 are two successive components of C both of which have left-unbounded edges and such that C_1 is above C_2. Then there is exactly one left-unbounded edge e of $V(L)$ lying vertically between C_1 and C_2 such that both its endpoints are closer to R and such that the faces above and below e contain left-unbounded segments of C_1 and C_2, respectively.*

A symmetric statement holds with "L" switched with "R" and "left" switched with "right".

Proof Omitted in this version but see Figure 3 for an illustration. □

This property determines for each edge $e \in C$ the points $q_i \in L$ and $q_j \in R$ such that e is a portion of $B(q_i, q_j)$:

Property 8 *Every edge $e \in C$ either leaves (according to the orientation of C induced by its linear order) or enters an edge of S or does both (when e is bounded) except when S is empty, a trivial case which can be handled separately.*

First assume e leaves edge $e_i \in S$ at some point c_i. If $e_i \in M$ and there exist edges in N less than e_i let e_{bi} be the greatest such edge. Then e is a portion of the bisector $B(lp(e_i), lp(e_{bi}))$. If $e_i \in M$ and there do not exist edges in N less than e_i let e_{min_N} be the least edge in N. Then e is a portion of the bisector $B(lp(e_i), rp(e_{min\_N}))$. If $e_i \in N$ a symmetric statement applies with "M" switched with "N".

Next assume e enters edge $e_i \in S$ *at some point* c_i. *If* $e_i \in M$ *and there exist edges in* N *greater than* e_i *let* e_{fi} *be the least such edge. Then e is a portion of the bisector* $B(rp(e_i), rp(e_{fi}))$. *If* $e_i \in M$ *and there do not exist edges in* N *greater than* e_i *let* e_{max_N} *be the greatest edge in* N. *Then e is a portion of the bisector* $B(rp(e_i), lp(e_{max\_N}))$. *If* $e_i \in N$ *a symmetric statement applies with "M" switched with "N".*

Proof In the case that e leaves $e_i \in M$ at point c_i and e_{bi} is the greatest edge in N preceding e_i, e_i must lie in the intersection of $F_L(lp(e_i))$ and $F_R(lp(e_{bi}))$ and so is a portion of $B(lp(e_i), lp(e_{bi}))$. The other cases may be similarly settled. □

Observe that, given both $S = \{e_1 < \ldots < e_r\}$ in sorted order and the edges of S that have unbounded edges of C leaving or entering them, we can, applying Property 8, fully construct C and determine the intersections of C with edges in S as an array $\{c_1 < \ldots < c_r\}$ sorted by y-coordinates (exploiting Property 2 which implies that the ordering of these intersections by y-coordinates is induced by the order of S).

3 Some Basic Parallel Algorithms

The basic operations that we shall require to do in parallel are *merging* of arrays, *list ranking*, and *prefix computation.* List ranking is, given a linked list L, to compute the distance each element of the list is from an end of the list and so to present L as an array. Prefix computation is, given a sequence $a_1, \ldots a_n$ of elements, to compute the partial sums $\oplus_{k=1}^{i} a_k$ for each i, $1 \leq i \leq n$, where $\oplus$ is a semigroup operation. Typical problems we shall solve by prefix computation are as follows: given a sequence $a_1, \ldots a_n$ of elements some of which are marked "red" find, for each a_i, (a) the number of red elements preceding it and (b) the largest index of a red element preceding it. Problem (a) may be solved by a prefix computation with the operation as addition and each red element representing a 1 and non-red element a 0; (b) may be solved by a prefix computation with the operation as max and each red element representing its index and non-red element representing $-\infty$. The parallel complexity of these operations is as described:

Property 9 *Merging of two arrays of total size* $O(n)$, *list ranking of a list of size* $O(n)$, *and prefix computation on a sequence of size* $O(n)$ *(represented as a linked list or array) may each be done in optimal* $O(\log n)$ *time on a CREW PRAM with* $n/\log n$ *processors.*

Proof For merging see [12] and for list ranking and prefix computation see [3]. □

4 A Parallel Algorithm for the Rectilinear Voronoi Diagram

Crucial to our Voronoi diagram algorithm is the following theorem:

Theorem 1 *Consider a partition of a set* P *containing* n *points into subsets* L *and* R *lying to the left and right of a vertical line* D. *Given the Voronoi diagrams* $V(L)$ *and* $V(R)$ *and each of* L, R, $Ver(L)$, *and* $Ver(R)$ *as an array of points sorted by* y*-coordinates one can compute the Voronoi diagram* $V(P)$ *and each of* P *and* $Ver(P)$ *as an array of points sorted by* y*-coordinates in* $O(\log n)$ *time on a CREW PRAM with* $n/\log n$ *processors.*

Proof The plan is to construct the dividing chain C and use Property 3 to "merge" $V(L)$ and $V(R)$. Accordingly, the following algorithm solves the problem in 8 steps. Henceforth, we shall assume, unless specified otherwise, that every point set carries an order specified by the y-coordinates of its points.

Step 1 Property 1 provides the geometric insight to replace planar point location by a merge followed by prefix operations:

Merge L, R, $Ver(L)$, and $Ver(R)$ into one array, say A.

Step 2 Add the points $(-\infty,-\infty)$ and $(\infty,-\infty)$ to the beginning of A and the points $(-\infty,\infty)$ and (∞,∞) to the end of A and mark them each as belonging to $Ver(R)$. With a constant number of prefix operations on A determine for each vertex in $Ver(L)$ of the form (x, y) where $x < \infty$ and for each point in L the first vertex in $Ver(R)$ of the form $(-\infty, y')$ preceding it and the first vertex in $V(R)$ of the form $(-\infty, y'')$ succeeding it. Further, determine for each vertex in $Ver(L)$ of the form (∞, y) the first vertex in $Ver(R)$ of the form (∞, y') preceding it and the first vertex in $V(R)$ of the form (∞, y'') succeeding it. This locates for each vertex in $Ver(L)$ and point in L the polygon of $V(R)$ to which it belongs and consequently determines its nearest vertex in R. Observe that for each vertex in $Ver(L)$, in fact for any point on an edge e of $V(L)$, a nearest point in L is trivially determined by the representation of $V(L)$: a nearest point will be q_i if e is a portion of the bisector $B(q_i, q_j)$

Step 3 Use Property 4 to determine M:

If the edge e has one end vertex closer to L and the other closer to R mark it as belonging to M. If the edge e has both ends closer to R decide if one of the rays emanating leftwards from q_i or q_j, where e is a portion of $B(q_i, q_j)$, intersects e. If not mark e and its endpoints as *deletable*. If one of the rays does intersect e (at most one can) at point m assume w.l.o.g. that it is the leftbound ray from q_i. Then, as is easily seen, m and q_i must belong to the same polygon of $V(R)$ and, since in Step 2 q_i has been located in some polygon of $V(R)$, we may locate m in the same polygon and determine its nearest vertex in R. If m is closer to L then split e into two semiedges at m and mark both as belonging to M. Henceforth, we shall not distinguish between semiedges and edges. On the other hand, if m is closer to R then mark e and its endpoints as deletable. This step may be done in constant time with one processor per edge of $V(L)$ and so in $O(\log n)$ time with $n/\log n$ processors.

Step 4 Order M using Properties 5 and 6:

By the representation of $V(L)$ every edge of each face of $V(L)$ has a pointer to its counterclockwise successor on that face. Perform a prefix operation on a counterclockwise list of edges of each face f to get a counterclockwise list of the edges bounding f that are in M. Comparing, in constant time per edge using Property 5, each edge with its successor and predecessor on this (circular) list locate the first and last edges according to the order in M. Then in constant time split the list between the first and last edges to get a linear list, the M-list of f (see Property 6). As there may be $O(n)$ lists on which to apply (parallel) prefix computation computing all the M-lists apparently takes $O(\log n)$ time with n processors but, the total size of all the lists being $O(n)$, we may apply Brent's principle [4] to reduce the processor requirement to $n/\log n$. An edge e in M which is the last on the M-list of face f_1 and the first on the M-list of face f_2 links the two M-lists. We still do not have all the links to represent M as a linked list. These "missing links" are due to the presence of components of C unbounded to the left. For example, in Figure 3, edge e_5 has not as yet acquired a pointer to its successor e_6 in M. We use Property 7 to resolve this difficulty. Each processor associated to a left-unbounded edge e of $V(L)$ with both endpoints closer to R links the last edge e_i of the M-list of the face below e to the first edge e_j of the M-list of the face above it. The processsor also marks e_i and e_j as having unbounded edges of C leaving and entering them, respectively. We now have all the links of M as a linked list. Finally perform list ranking to get M as a sorted array.

Step 5 Repeat steps symmetric to Steps 2–4 for $V(R)$ to obtain the sorted array N.

Step 6 Merge M and N to obtain the array $S = \{e_1 < \ldots < e_r\}$ using Property 5 to make constant time comparisons between edges from M and N.

Step 7 Use Property 8 to determine C and its intersections with edges in S:

With S sorted and all edges of S incident to unbounded edges of C being marked (marking was done in Steps 4 and 5 and, further, mark the first and last edges of S as having unbounded edges of C entering and leaving them, respectively), perform a constant number of prefix operations to determine the edges e_{bi} and e_{fi} corresponding to each edge $e_i \in S$ and determine e_{min_M}, e_{min_N}, e_{max_M}, and e_{max_N}. Then apply the method of Property 8 to retrieve C and its intersections with S as a sorted array $\{c_1 < \ldots < c_r\}$.

Step 8 Finally, to produce the desired output first delete all the edges marked as deletable from $V(L) \cup V(R)$ and all vertices marked as deletable from the array $Ver(L) \cup Ver(R)$ formed from merging $V(L)$ and $V(R)$. Then for each edge e_i in M (N) delete the vertex closer to R (L) and shorten e_i to the point c_i of its intersection with C. All deletions from a sorted array or list may be done by first counting for each element, with two prefix operations, the number of elements preceding it and the number of elements succeeding it that are to be deleted and accordingly repositioning the undeleted elements in constant time. Deletions change the array $\underline{Ver(L)} \cup Ver(R)$ to the array $\overline{Ver(P)}$. Now merge the array $\{c_1 < \ldots < c_r\}$ with $\overline{Ver(P)}$ to obtain the sorted array $Ver(P)$. To complete $V(P)$ add to it the edges in C doing n processor and constant time ($= n/\log n$ processor and $O(\log n)$ time) bookkeeping to obtain the new representation of $V(P)$ as a plane graph. Merge L and R to obtain P as a sorted array.

Observe that, with the parallel complexity of the operations described in Property 9, the algorithm takes $O(\log n)$ time on a CREW PRAM with $n/\log n$ processors. □

We may now use Theorem 1 in a divide-and-conquer algorithm to compute the rectilinear Voronoi diagram. However, as $n/\log n$ is a subadditive function of n, this is not quite straightforward and some care has to be taken in the recursion.

Imagine the points of P sorted by x-coordinates and at the leaves of a complete binary tree of height $\log n$ (assume, without loss, that n is a power of 2). The plan is to merge up this tree computing as we ascend, for each node, the Voronoi diagram of the leaves of the subtree rooted at it. We have at our disposal $n/\log n$ processors. Number the levels of the tree from 0 at the leaves to $\log n$ at the root. Now upto level $\log\log n - 1$ (at and below this level the number of merges to be performed at each level is at least as large as the number of processors available) use an optimal sequential algorithm after distributing the merge problems at each level equally to the $n/\log n$ processors. The sequential algorithm to merge, as for example described by Lee and Wong [11] (with corrections as their algorithm has minor flaws), takes time linear in the size of the problem even when modified to carry the extra order structure described in Theorem 1. This means spending $O(\log n)$ time at each level till level $\log\log n - 1$ for a total of $O(\log n \log\log n)$ time. From level $\log\log n$ upwards at each level the number of processors exceeds the number of merges to be done and we can use Theorem 1. To do so first observe that from Theorem 1 it may be deduced that, with $p \leq n/\log n$ processors, we can solve the problem of that theorem in time $O(n/p)$. Now at each level i, $\log\log n \leq i \leq \log n - 1$, the number of processor available per merge problem is $2^{i-\log\log n+1}$ and the size of each merge problem is 2^{i+1}. As $2^{i-\log\log n+1} \leq 2^{i+1}/\log 2^{i+1}$, for $\log\log n \leq i \leq \log n - 1$, apply the preceding observation to spend $O(2^{i+1}/2^{i-\log\log n+1}) = O(\log n)$ time at each level for a total of $O(\log^2 n)$ time to ascend to the root.

This proves the main theorem:

Theorem 2 *The rectilinear Voronoi diagram of a set P containing n points may be computed in $O(\log^2 n)$ time on a CREW PRAM with $n/\log n$ processors.* □

5 Conclusions

We have described an optimal parallel algorithm to compute the Voronoi diagram under the rectilinear metric that takes $O(\log^2 n)$ time using $n/\log n$ processors. An intriguing question is whether $O(\log n)$ time is possible with n processors using the fractional cascading methods of Atallah, Cole, and Goodrich [2]. This seems to be difficult as a problem with cascading here is that each level of recursion throws up new Voronoi points which must be anticipated and incorporated efficiently into the cascade.

Another question is whether our methods may be applied to the Euclidean case by exploiting the geometry to impose some sort of linear order (maybe only locally) on the planar regions of the Euclidean Voronoi diagram.

Acknowledgements

I would like to thank my supervisor Professor Quentin Stout for many helpful discussions and my colleague Phil MacKenzie for pointing out some important references.

References

[1] A. Aggarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing, C. Yap, "Parallel computational geometry," *Algorithmica*, **3** (1988), 293-327.

[2] M. J. Atallah, R. Cole, M. T. Goodrich, "Cascading divide-and-conquer: a technique for designing parallel algorithms," *Proc. 28th IEEE Symp. on FOCS*, 1987, 151-160.

[3] R. J. Anderson, G. L. Miller, "Deterministic parallel list ranking," *Lecture Notes in C.S. 319: 3rd AWOC*, Springer Verlag, 1988, 81-90.

[4] R. P. Brent, "The parallel evaluation of general arithmetic expressions," *J. ACM*, **21** (1974), 201-206.

[5] K. Q. Brown, "Voronoi diagrams from convex hulls," *Inform. Process. Lett.*, **9** (1979), 223-228.

[6] A. Chow, "Parallel algorithms for geometric problems," Ph.D. Dissertation, University of Illinois at Urbana-Champaign, 1980.

[7] R. Cole, "Parallel merge sort," *Proc. 27th IEEE Symp. on FOCS*, 1986, 511-516.

[8] F. K. Hwang, "An $O(n \log n)$ algorithm for rectilinear minimal spanning trees," *J. ACM*, **26** (1979), 177-182.

[9] C. S. Jeong, D. T. Lee, "Parallel geometric algorithms on a mesh-connected computer," *Algorithmica*, **5** (1990), 155-177.

[10] D. T. Lee, "Two-dimensional Voronoi diagrams in the L_p-metric," *J. ACM*, **27** (1980), 604-618.

[11] D. T. Lee, C. K. Wong, "Voronoi diagrams in L_1 (L_∞) metrics with 2-dimensional storage applications," *SIAM J. Comp.*, **9** (1980), 200-211.

[12] C. P. Kruskal, "Searching, merging, and sorting in parallel computation," *IEEE Trans. Comp.*, **C-32** (1983), 942-946.

[13] F. P. Preparata, M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[14] M. I. Shamos, "Computational geometry," Ph.D. Dissertation, Yale University, 1977.

[15] M. I. Shamos, D. Hoey, "Closest point problems," *Proc. 16th IEEE Symp. on FOCS*, 1975, 151-162.

[16] P. Widmayer, Y. F. Wu, C. K. Wong, "Distance problems in computational geometry with fixed orientations," *Proc. 1st ACM Symp. on Comp. Geom.*, 1985, 186-195.

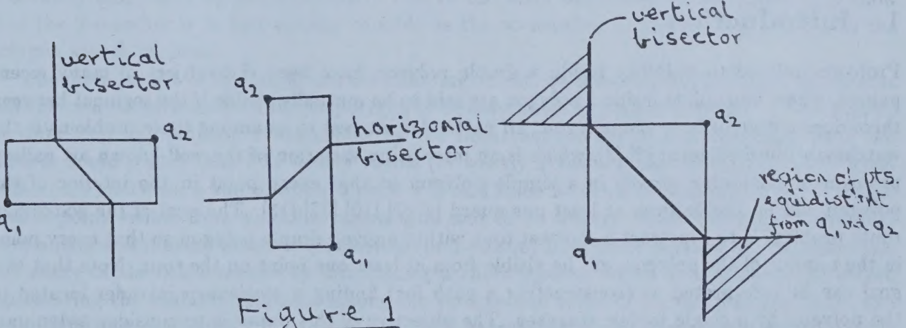

vertical bisector
q_2
q_1
q_2
horizontal bisector
q_1
vertical bisector
q_2
region of pts. equidist'nt from q_1 & q_2
q_1

Figure 1

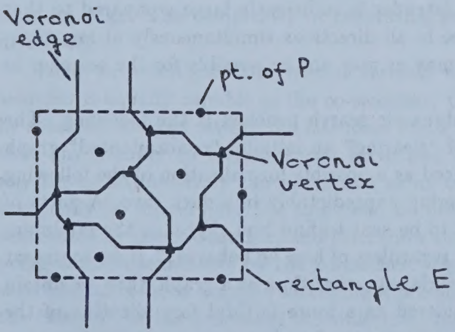

Voronoi edge
pt. of P
Voronoi vertex
rectangle E

Figure 2 (from [11])

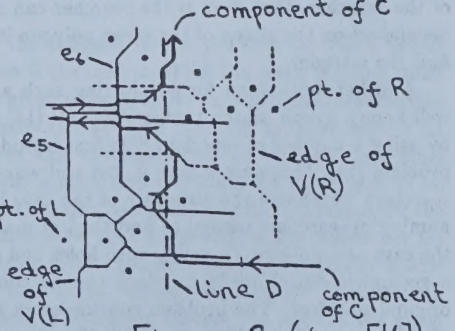

component of C
e_6
pt. of R
e_5
edge of V(R)
pt. of L
edge of V(L)
line D
component of C

Figure 3 (from [16])

The Polygon Search Problem

David Crass † Ichiro Suzuki † Masafumi Yamashita ‡

† Department of Electrical Engineering and Computer Science
University of Wisconsin–Milwaukee, Milwaukee, WI 53201
‡ Department of Electrical Engineering, Faculty of Engineering
Hiroshima University, Higashi-Hiroshima 724, Japan

Abstract The problem of searching for a mobile intruder in a simple polygon by a single mobile searcher is considered. We investigate the capabilities of searchers having different degrees of visibility by introducing the searcher having k flashlights whose visibility is limited to k rays emanating from his position, and the searcher having a point light source who can see in all directions simultaneously. We present necessary conditions and sufficient conditions for a polygon to be searchable by various searchers. We also discuss two special cases of the problem in which the searcher having two flashlights is as capable as the searcher having a point light source, and give a simple necessary and sufficient condition for a given polygon to be searchable by the searcher having two flashlights for each of the two cases. The complexity of generating a search schedule under many of these conditions is also discussed. Many of the results are proved by using an abstract structure called a chord system which represents the visibility relations among the vertices and edges of a given polygon.

1 Introduction

Problems related to visibility inside a simple polygon have been the subject of many recent papers, where two points inside a polygon are said to be mutually visible if the segment between them does not go outside the polygon. Of particular interest to us among these problems is the watchman route problem [2] [3], which is an interesting variation of the well-known art gallery problem of stationing guards in a simple polygon so that every point in the interior of the polygon will be visible from at least one guard [4] [7] [10] [12] [13]. The goal of the watchman route problem is to construct a shortest tour within a given simple polygon so that every point in the interior of the polygon will be visible from at least one point on the tour. Note that this goal can be interpreted as (constructing a path for) finding a *stationary* intruder located in the polygon by a single mobile searcher. The objective of this paper is to consider a dynamic version of the watchman route problem in which the intruder is assumed to be *mobile* rather than stationary. Obviously, if the speed of the intruder is sufficiently large compared to that of the searcher, then even if the searcher can see in all directions simultaneously at any time, depending on the shape of the given polygon it may or may not be possible for the searcher to find the intruder.

Another motivation for considering such a dynamic search problem is the following. The well-known graph search problem [9] [11] [14] of "clearing" an initially "contaminated" graph by using a number of searchers was first introduced as a possible formalization of the following problem [14]: "Suppose a man is lost and wandering unpredictably in a dark cave. A party of searchers who know the structure of the cave is to be sent to find him. What is the minimum number of searchers needed to find the lost man regardless of how he behaves?" If we represent the cave as a polygon (possibly with holes and overlaps) rather than as a graph, then we obtain a geometric search problem which can be considered as a more faithful formalization of the original problem. The problem considered in this paper can be viewed as a restricted version of this geometric problem in which the given polygon is simple and the number of available searchers is one.

We refer to the dynamic search problem informally introduced above as the *polygon search problem*. In this problem we represent both the searcher and the intruder as a point which can move continuously within a given polygon, and assume that the intruder can move arbitrarily faster than the searcher. To investigate the capabilities of the searchers having different degrees of visibility, we introduce the k-searcher for each integer $k \geq 1$ and the ∞-searcher. The k-searcher is the searcher having k "flashlights" whose visibility is limited to k rays emanating from his position, where the directions of the rays can be changed continuously with bounded angular rotation speed. The ∞-searcher is the searcher having the visibility of 360 degrees who can see in all directions simultaneously at any time. Given a simple polygon, the objective here is to decide whether there exists a "schedule" for the given searcher to find the intruder regardless of how he moves, and if so, to generate such a schedule. Detection of a mobile intruder in a simple polygon was first considered in the searchlight scheduling problem [15] in which the rays of stationary "searchlights" are used to find the intruder, but to our knowledge, no concrete results on the polygon search problem have been reported.

We say that a polygon is searchable by a given searcher if there exists a schedule for the searcher for finding the intruder regardless of how he moves. Note that any polygon searchable by the k-searcher is searchable by the k'-searcher for any $k' > k$ and also by the ∞-searcher. Here we can show that (1) the 2-searcher is strictly more capable than the 1-searcher (i.e., there exists a polygon searchable by the 2-searcher but not by the 1-searcher), and (2) any n-sided polygon searchable by the ∞-searcher is searchable by the $(\lfloor n/2 \rfloor + 1)$-searcher. Thus a fundamental question here is whether there exists an n-sided polygon searchable by the k-searcher but not by the $(k-1)$-searcher for some $3 \leq k \leq \lfloor n/2 \rfloor + 1$. So far we have not been able to find such a polygon, and we conjecture that any polygon searchable by the ∞-searcher is actually searchable by the 2-searcher. One of the main objectives of this paper is to show that the 2-searcher is in fact equally capable as the ∞-searcher in two special versions of the polygon search problem.

The main results of this paper are the following. First, we present some necessary conditions and sufficient conditions for a simple polygon to be searchable by various searchers. For example, we show that any polygon P searchable by the 1-searcher (or the ∞-searcher) must have two vertices u and v such that P is weakly visible (or weakly 2-visible) from the Euclidean shortest path $\pi(u, v)$ between u and v, where a point x is said to be 2-visible from a point y if x is visible from a point visible from y, and a region Q is said to be weakly visible (or weakly 2-visible) from a region R if every point in Q is visible (or 2-visible) from some point in R. We prove this and many of the other results by introducing a *chord system*, which is an abstract structure representing the visibility relations among the vertices and edges of a given polygon. Also, two of the sufficient conditions state that a polygon is searchable by the 2-searcher if either (1) it is weakly visible from $\pi(u, v)$ for some points u and v on its boundary, or (2) it is weakly 2-visible from an edge. The complexity of generating search schedules under these conditions are also discussed.

Then we investigate two special versions of the polygon search problem in which the 2-searcher is equally capable as the ∞-searcher. One is the case in which the given polygon must be searched without letting the intruders reach a given edge u, under an additional assumption that any number of intruders can leave and enter the polygon through another edge v at any time. Intuitively, v can be viewed as an open exit through which all the intruders must eventually be forced out. For this case, we present a necessary condition for a polygon to be searchable by the ∞-searcher, and then show that the same condition is in fact sufficient for the polygon to be searchable by the 2-searcher. We also show that a schedule of the 2-searcher for searching an n-sided polygon satisfying this condition can be generated in $O(n^2 \log n)$ time, and the "length" of the generated schedule (i.e., the number of "elementary actions" it contains), which is $O(n^2)$, is asymptotically worst case optimal. The second special case is when the given polygon is a *hedgehog*, which is a simple polygon consisting of a convex body and narrow hooked pins (corridors) (see Figures 3 and 4). We show that any hedgehog searchable by the ∞-searcher is also searchable by the 2-searcher, and that a schedule of the 2-searcher of length $O(m)$ for searching a given n-sided hedgehog having m hooked pins can be generated in $O(n + m \log n)$ time.

The polygon search problem is stated formally in Section 2. The results related to the necessary conditions and sufficient conditions are presented in Sections 3 and 4, respectively. The two special versions of the problem are discussed in Section 5 and 6. Concluding remarks are found in Section 7. Due to space limitation we are unable to present any of the proofs in this extended abstract. All the missing details can be found in the full paper.

2 Problem Formulation

Let P be a simple polygon.[1] Two points x and $y \in P$ are said to be mutually *visible* if $\overline{xy} \subseteq P$. We let $V(x)$ denote the set of points in P which are visible from x.

Let us introduce the searcher who can walk within P holding k flashlights, where k is a positive integer and a flashlight emits a single light beam which is blocked as soon as it intersects with the exterior of P. We call this searcher the *k-searcher*. Formally:

Definition 1 A *schedule* of the k-searcher for P is a tuple $\mathcal{S} = \langle S, F_1, F_2, \ldots, F_k \rangle$ of $k+1$ continuous functions $S : [0,T] \to P$ and $F_1, F_2, \ldots, F_k : [0,T] \to \mathcal{R}$, where $[0,T]$ is an interval of real time and $\mathcal{R}$ is the set of real numbers. A point $x \in P$ is *illuminated* at time $t \in [0,T]$ during the execution of $\mathcal{S}$ iff x is on the intersection of $V(S(t))$ and the k rays emanating from $S(t)$ in the directions $F_1(t), F_2(t), \ldots, F_k(t)$, respectively.

$S(t)$ is the location of the searcher and $F_1(t), F_2(t), \ldots, F_k(t)$ are the directions of the k flashlights at time t, respectively, where directions are measured in radian counterclockwise from the positive x-axis. Intuitively, the illuminated points are those that the k-searcher can "see" at any given time. Note that the visibility of the k-searcher is limited to k rays emanating from his position.

Let us introduce another type of searcher who can see in all directions simultaneously. We call this searcher the *∞-searcher*, since we can imagine that this searcher has a point light source which can be viewed as a collection of infinitely many flashlights aimed in all directions.

Definition 2 Let P be a simple polygon. A *schedule* of the ∞-searcher for P is $\mathcal{S} = \langle S \rangle$, where $S : [0,T] \to P$ is any continuous function over an interval $[0,T]$ of real time. A point $x \in P$ is said to be *illuminated* at time $t \in [0,T]$ during the execution of $\mathcal{S}$ iff $x \in V(S(t))$.

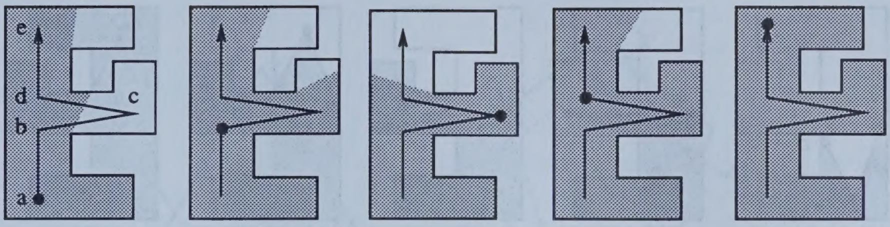
$S(t)$ is the location of the ∞-searcher at time t. Again, the illuminated points are those that the ∞-searcher can "see" at any given time. Note that the set of illuminated points at time t coincides with the set of points which are visible from $S(t)$.

For points x and $y \in P$, if at time t during the execution of a schedule there exists a path within P between them not containing any illuminated point, then the intruder can move from x to y instantaneously without being seen by the searcher. Also, initially the intruder can be located at any position which the searcher cannot see. This concept is made precise in the following definition.

Definition 3 Let $\mathcal{S}$ be a schedule of any searcher defined over the interval $[0,T]$ of real time. A point $x \in P$ is said to be *contaminated* at time $t \in [0,T]$ during the execution of $\mathcal{S}$ iff there exist points $x_0, x_1, \ldots, x_m \in P$ and times $0 = t_0 \leq t_1 \leq \ldots \leq t_m \leq t_{m+1} = t$ such that (1) $x = x_m$, (2) for each $0 \leq i \leq m$, x_i is not illuminated at any time in $[t_i, t_{i+1}]$, and (3) for each $1 \leq i \leq m$, at time t_i there exists a path within P between x_{i-1} and x_i not containing any illuminated point. A point which is not contaminated is said to be *clear*. A region $R \subseteq P$ is said to be *contaminated* if it contains a contaminated point; otherwise it is *clear*.

---

[1] The term *simple polygon* is used to denote the union of a closed simple polygonal chain and its interior.



(a) (b) (c) (d) (e)

Figure 1: A polygon and a search schedule of the ∞-searcher.

Intuitively, $x \in P$ is contaminated at time $t \in [0,T]$ iff the intruder who can move continuously with unbounded speed and who has not been seen by the searcher in the interval $[0,t]$ can be located at x at t. Note that at time zero, a point is contaminated iff it is not illuminated. Also, at any time an illuminated point is clear, and a contaminated point remains contaminated until it becomes illuminated. Since our objective is to find the intruder regardless of how he moves, we have:

Definition 4 Let $\mathcal{S}$ be a schedule of the k-searcher (or ∞-searcher) for P defined over the interval $[0,T]$ of real time. S is called a *search schedule* for P if P is clear at time T. P is said to be *k-searchable* (or *∞-searchable*) if there exists a search schedule of the k-searcher (or ∞-searcher) for P.

Since the concept of contaminated points given in Definition 3 is based on the assumption that the intruder can move arbitrarily faster than the searcher, we can easily show that the "speed" at which the searcher "executes" a given schedule is irrelevant to whether or not P is clear at the end of the execution. Thus we can represent a schedule $\mathcal{S}$ of any searcher simply as a sequence of some elementary actions, instead of giving formal descriptions of the functions contained in $\mathcal{S}$. In particular, the only elementary action of the ∞-searcher which we will use in this paper is to move over a given line segment, and hence a schedule of the ∞-searcher can simply be given as a directed polygonal chain within P. For the k-searcher, the elementary actions we use are (1) to aim a flashlight at a given point, (2) to rotate a flashlight either clockwise or counterclockwise to illuminate a given point, and (3) to move over a segment aiming the flashlights at given points. Obtaining a formal description of a schedule as a collection of functions from a sequence of such actions is straightforward. We define the *length* of a schedule to be the number of elementary actions it contains.

Example 1 Figure 1 shows a simple polygon and the directed polygonal chain $\overline{abcde}$ determined by a search schedule of the ∞-searcher. The chain consists of four segments, and hence the length of the schedule is four. The regions which are clear when the ∞-searcher reaches a, b, c, d and e are shown shaded in Figure 1(a), (b), (c), (d) and (e), respectively (the ∞-searcher is shown by $\bullet$). □

Example 2 Figure 2 shows how the polygon of Example 1 can be cleared by the 2-searcher. Again, clear regions are shown shaded. Initially the two flashlights are aimed at a, and then one of them is rotated counterclockwise until it is aimed at b (Figure 2(a)). Then the 2-searcher moves to a new position aiming the two flashlights continuously at a and b, respectively (Figure 2(b)). This movement requires two consecutive moves over a segment, since the positions of the 2-searcher in Figure 2(a) and (b) are not mutually visible. The rest of the operation should be self-explanatory. Note that only the elementary actions introduced above are used. □

Some points in the polygon may have to be recontaminated repeatedly during the search, as the following example shows.

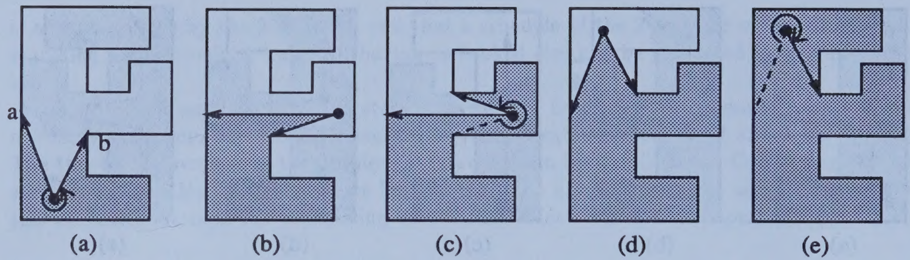


Figure 2: Clearing the polygon of Example 1 using the 2-searcher.

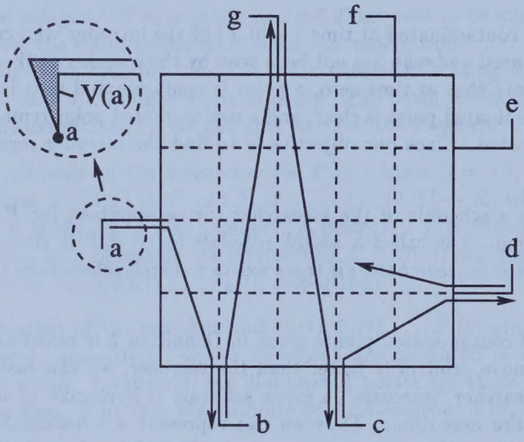


Figure 3: A square with seven hooked pins.

Example 3 The polygon shown in Figure 3 consists of a square and seven narrow hooked pins, where each pin is drawn schematically as a polygonal chain. Each pin is assumed to be sufficiently narrow so that the visibility (into the square) from the points near the bend is limited approximately to the neighborhood of the corresponding dotted line. Let $a, b, \ldots,$ and g be the vertices at the "tip" of the pin, as is shown in the figure. Note that the ∞-searcher must visit each of $V(a)$, $V(b)$, …, and $V(g)$ to clear these tips. This polygon is ∞-searchable, since it can be cleared by the ∞-searcher who visits $V(a)$, $V(b)$, $V(g)$, $V(c)$, $V(d)$, $V(a)$, $V(e)$, $V(f)$, $V(c)$, $V(g)$, $V(b)$ and $V(a)$ in this order by using the shortest paths to go from one region to the next. (An initial segment of this path is shown in Figure 3.) The reader can easily verify that any search schedule for this polygon must start from and end at a point in $V(a)$ (unless the ∞-searcher makes some "useless" moves before and/or after the execution of such a schedule), and a must be recontaminated twice during the search. It is also easy to see that this polygon can be cleared by the 2-searcher who moves along the path of the ∞-searcher described above. □

Example 4 Figure 4 shows schematic drawings of two polygons which are not ∞-searchable. The one shown in Figure 4(a) consists of a triangle and three narrow hooked pins, where again each pin is drawn as a polygonal chain with a bend. To see that this polygon is not ∞-searchable, note that whenever one of the three tips a, b and c is illuminated by the ∞-searcher, there exists a path between the remaining two tips not containing any illuminated point. This, together with the fact that initially at most one of the three tips can be illuminated, implies that at most one of the three tips can be clear at any time during the search by the ∞-searcher.

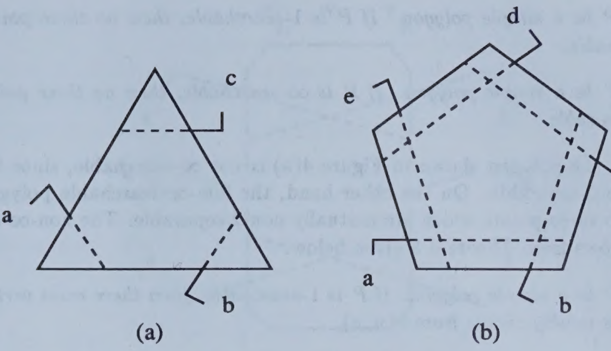


Figure 4: Two polygons which are not ∞-searchable.

Thus this polygon is not searchable by the ∞-searcher. The polygon shown in Figure 4(b) is a "windmill" consisting of a convex body and five narrow hooked pins. Note that initially at most one of the five tips a, b, c, d and e can be clear. Thus, without loss of generality let a be the tip which becomes clear first among the five. If any of b, c and d is cleared next, then that tip will be the only clear tip since a becomes recontaminated. It is possible to clear e without recontaminating a, but both become recontaminated if b or c is cleared next, and d and e will be the only clear tips if d is cleared next. Therefore at most two of the five tips can become clear simultaneously, and hence there exists no search schedule for this polygon. □

By definition, any simple polygon which is k-searchable is k'-searchable for any $k' > k$ and ∞-searchable. Also, the 2-searcher is strictly more capable than the 1-searcher since (we can show that) the 2-searchable polygon of Figure 2 is not searchable by the 1-searcher. Furthermore, we can prove the following theorem by showing that the $(\lfloor n/2 \rfloor + 1)$-searcher can "simulate" the ∞-searcher.

Theorem 1 *If an n-sided simple polygon is ∞-searchable, then it is $(\lfloor n/2 \rfloor + 1)$-searchable.*

Thus we wish to know whether there exists an n-sided simple polygon which is k-searchable but not $(k-1)$-searchable for some $3 \leq k \leq \lfloor n/2 \rfloor + 1$. So far we have not been able to find such a polygon, and we conjecture that any ∞-searchable polygon is actually 2-searchable. In Sections 5 and 6 we show that this indeed is the case for two special version of the polygon search problem.

3 Necessary Conditions

Let P be a simple polygon. For a point $x \in P$, define $V^2(x) = \bigcup_{y \in V(x)} V(y)$. If $y \in V^2(x)$, then we say that y is *2-visible* from x. For regions Q and $R \subseteq P$, we say that Q is *weakly visible* (or *weakly 2-visible*) from R if every point in Q is visible (or 2-visible) from some point in R. For points x, y and $z \in P$, y and z are said to be *separable* (or *2-separable*) from x if every path within P between y and z contains at least one point in $V(x)$ (or $V^2(x)$). Note that since P is simple, y and z are separable (or 2-separable) from x iff $\pi(y,z)$ contains at least one point in $V(x)$ (or $V^2(x)$), where $\pi(y,z)$ is the Euclidean shortest path within P between y and z. (For example, in Figure 3, b and g are 2-separable from a, but b and c are not.) Finally, points x, y and $z \in P$ are said to be *mutually non-separable* (or *mutually non-2-separable*) if no two points out of the three are separable (or 2-separable) from the third. The following theorems are immediate from definition and discussions similar to that given in Example 4.

Theorem 2 *Let P be a simple polygon. If P is 1-searchable, then no three points in P are mutually non-separable.*

Theorem 3 *Let P be a simple polygon. If P is ∞-searchable, then no three points in P are mutually non-2-separable.*

By Theorem 3, the polygon shown in Figure 4(a) is not ∞-searchable, since tips a, b and c are mutually non-2-separable. On the other hand, the non-∞-searchable polygon shown in Figures 4(b) has no three points which are mutually non-2-separable. The non-∞-searchability of this polygon follows from Theorem 5 given below.

Theorem 4 *Let P be a simple polygon. If P is 1-searchable, then there exist vertices u and v of P such that P is weakly visible from $\pi(u, v)$.*

Theorem 5 *Let P be a simple polygon. If P is ∞-searchable, then there exist vertices u and v of P such that P is weakly 2-visible from $\pi(u, v)$.*

These two theorems are proved by using *chord systems*, which are defined as a set of chords of the unit circle representing the visibility relations among the vertices and edges of the given polygon. The details are found in the full paper.

4 Sufficient Conditions

Theorem 6 *Let P be a simple polygon which is weakly visible from an edge $\overline{uv}$. Then P is 1-searchable. Furthermore, P can be cleared by the 1-searcher who moves only within $\overline{uv}$.*

Theorem 7 *Let P be a simple polygon which is weakly 2-visible from an edge $\overline{uv}$. Then P is 2-searchable.*

Using the results given in [1] [6] [17], we can generate a search schedule of length $O(n)$ for an n-sided polygon satisfying the conditions of Theorems 6 and 7 in $O(n)$ time and $O(n \log \log n)$ time, respectively.

Let us denote by ∂P the boundary of P.

Theorem 8 *Let P be a simple polygon which is weakly visible from $\pi(u, v)$ for some points u and $v \in \partial P$. Then P is 2-searchable.*

Clearly, any polygon P satisfying the condition of Theorem 8 is ∞-searchable. For example, we can clear such P by simply moving the ∞-searcher along the path determined by an arbitrary search schedule of the 2-searcher, ignoring the instructions for rotating the flashlights. The following theorem shows that the search schedule for the ∞-searcher can sometimes be shorter than that obtained by this conversion.

Theorem 9 *Let P be a simple polygon which is weakly visible from $\pi(u, v)$ for some points u and $v \in \partial P$. Then $\pi(u, v)$, viewed as a directed path from u to v, is a search schedule of the ∞-searcher for P.*

As the following theorem shows, points u and v satisfying the condition of Theorem 9 can be found efficiently, if they exist. We can find these points by first transforming certain "chords" of a polygon [8] into a set of chords of a unit circle, and then using the $O(n \log n)$ time stabbing line algorithm of [5].

Theorem 10 *For an n-sided simple polygon P, whether there exist points u and $v \in \partial P$ such that P is weakly visible from $\pi(u, v)$ can be tested and, if so, a pair of such u and v can be obtained in $O(n \log n)$ time.*

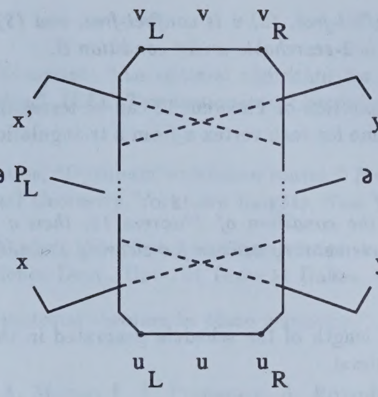


Figure 5: x and y are in conflict with respect to u; x' and y' are in conflict with respect to v.

5 A Necessary and Sufficient Condition for a Restricted Case

As is observed in Example 3, one of the reasons for the difficulty of deciding whether a given polygon P is k-searchable or ∞-searchable is that some vertices and edges of P may have to be recontaminated repeatedly during the search. This observation suggests us to consider a restricted version of the polygon search problem in which some vertices or edges of P must remain clear during the search. Specifically, the problem we consider in this section is the following.

> Given P and two edges u and v of P, clear P under the following condition $\mathcal{B}$: (1) u must remain clear throughout the search, and (2) at any time, any point on v which is not illuminated is considered to be contaminated.

A possible interpretation of this requirement is that (1) v represents an open exit between P and its exterior through which any number of intruders can leave and enter P at any time, and (2) the searcher must force all the intruders out of P through v without allowing any of them to reach u.

In the rest of this section, let P be an n-sided simple polygon, and let $u = \overline{u_L u_R}$ and $v = \overline{v_L v_R}$ be the special edges described above, where u_L, v_L, v_R and u_R appear in this order clockwise in ∂P. We let ∂P_L denote the portion of ∂P from u_L to v_L taken clockwise (i.e., the "left" boundary of P from u_L to v_L). Similarly, we let ∂P_R denote the portion of ∂P from u_R to v_R taken counterclockwise (i.e., the "right" boundary of P from u_R to v_R). For vertices $x \in \partial P_L$ and $y \in \partial P_R$, we say that x and y are in *conflict* with respect to u if (1) u_L and x are not 2-separable from y and (2) u_R and y are not 2-separable from x. Similarly, x and y are said to be in *conflict* with respect to v if (1) v_L and x are not 2-separable from y and (2) v_R and y are not 2-separable from x. See Figure 5. We say that u (or v) is *conflict-free* if there do not exist such vertices which are in conflict with respect to it. Finally, we say that u and v *satisfy the weak 2-visibility condition* if ∂P_L is weakly 2-visible from $\pi(u_R, v_R)$ and ∂P_R is weakly 2-visible from $\pi(u_L, v_L)$.

Theorem 11 *If P is ∞-searchable under condition $\mathcal{B}$, then (1) u is conflict-free, (2) v is conflict-free, and (3) u and v satisfy the weak 2-visibility condition.*

The following theorem, together with Theorem 11 and the fact that any 2-searchable polygon is ∞-searchable, shows that the condition given in Theorem 11 is in fact necessary and sufficient for P to be searchable under condition $\mathcal{B}$ by the ∞-searcher and the k-searcher for any $k \geq 2$.

Theorem 12 *If (1) u is conflict-free, (2) v is conflict-free, and (3) u and v satisfy the weak 2-visibility condition, then P is 2-searchable under condition $\mathcal{B}$.*

Whether P satisfies the condition of Theorem 12 can be tested in $O(n^2)$ time, basically by constructing $V^2(x)$ in $O(n)$ time for each vertex x from a triangulation of P [16]. Furthermore, we have:

Theorem 13 *If P satisfies the condition of Theorem 12, then a search schedule of the 2-searcher consisting of $O(n^2)$ elementary actions for clearing P under condition $\mathcal{B}$ can be generated in $O(n^2 \log n)$ time.*

It can be shown that the length of the schedule generated in the proof of Theorem 12 is asymptotically worst-case optimal.

6 Hedgehogs

A *hedgehog* is a simple polygon consisting of a convex body and a set of narrow hooked pins (corridors). Examples of hedgehogs are shown in Figures 3 and 4. We require that the pins are sufficiently narrow and appropriately oriented so that no point in the entrance of a pin is 2-visible from the tip of any other pin. The following theorem is immediate.

Theorem 14 *Let P be a hedgehog having m hooked pins. Then P is 1-searchable iff $m \leq 2$.*

The 2- and ∞-searchability of a hedgehog can be determined by examining the 2-separability of the tips of two hooked pins from the tip of a third pin, over all combinations of three pins. This 2-separability relation can be represented by a chord system similar to that for general polygons mentioned in Section 3. It turns out that using this chord system, we can state a necessary condition for a hedgehog P to be ∞-searchable, and then show that the same condition is in fact sufficient for P to be 2-searchable. (The condition can be found in the full paper.) By this observation, we obtain the following theorem.

Theorem 15 *Let P be a hedgehog. P is ∞-searchable iff P is 2-searchable.*

As for the complexity of generating a search schedule, we have:

Theorem 16 *Let P be an n-sided hedgehog having m hooked pins. Whether P is 2-searchable can be tested and if so, a search schedule of the 2-searcher consisting of $O(m)$ elementary actions for clearing P can be generated in $O(n + m \log n)$ time.*

7 Concluding Remarks

We have posed the polygon search problem and presented some necessary conditions and sufficient conditions for a polygon to be searchable by various searchers. We have also shown that the 2-searcher having two flashlights is as capable as the ∞-searcher who can see in all directions simultaneously in two special versions of the problem. An interesting open problem is to prove or disprove the conjecture that the 2-searcher can search any polygon searchable by the ∞-searcher. This problem, as well as another major open problem of determining the complexity of deciding whether a given polygon is searchable by a given searcher in the general case, is suggested for future research.

References

[1] D. Avis and G. T. Toussaint, "An optimal algorithm for determining the visibility of a polygon from an edge," *IEEE Transactions on Computers C-30*, December 1981, pp. 910–914.

[2] W.-P. Chin and S. Ntafos, "Optimum watchman routes," *Proc. of the 2nd Annual Symposium on Computational Geometry*, Yorktown Heights, New York, June 1986, pp. 24–33.

[3] W.-P. Chin and S. Ntafos, "Shortest watchman routes in simple polygons," Technical Report, Computer Science Dept., Univ. of Texas at Dallas, 1987.

[4] V. Chvátal, "A combinatorial theorem in plane geometry," *J. Combinatorial Theory (B) 18*, 1975, pp. 39-41.

[5] H. Edelsbrunner, H. A. Mauer, F. P. Preparata, A. Rosenberg, E. Welzl and D. Wood, "Stabbing line segments," *BIT 22*, 1982, pp. 274–281.

[6] L. Guibas, J. Hershberger, D. Leven, M. Sharir and R. Tarjan, "Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons," *Algorithmica 2*, 1987, pp. 209–233.

[7] J. Kahn, M. Klawe and D. Kleitman, "Traditional galleries require fewer watchmen," *SIAM J. Algebraic and Discrete Methods 4*, No. 2, 1983, pp. 194–206.

[8] Y. Ke, "Polygon visibility algorithms for weak visibility and link distance problems," Ph.D. Dissertation, Department of Computer Science, Johns Hopkins University, 1989.

[9] A. S. LaPaugh, "Recontamination does not help to search a graph," TR-335, Dept. of Electrical Engineering and Computer Science, Princeton University, 1985.

[10] D. T. Lee and A. K. Lin, "Computational complexity of Art Gallery problems," *IEEE Transactions on Information Theory IT-32*, No. 2, 1986, pp. 276–282.

[11] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson and C. H. Papadimitriou, "The complexity of searching a graph," *J. ACM 35*, No. 1, January 1988, pp. 18–44.

[12] J. O'Rourke, "An alternate proof of the rectilinear Art Gallery theorem," *J. Geometry 21*, 1983, pp. 118–130.

[13] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, 1987.

[14] T. D. Parsons, "Pursuit-evasion in a graph," in *Theory and Applications of Graphs*, Y. Alavi and D. Lick, eds., Lecture Notes in Mathematics 642, Springer-Verlag, Berlin, 1976, pp. 426–441.

[15] K. Sugihara, I. Suzuki and M. Yamashita, "The searchlight scheduling problem," to appear in *SIAM J. Computing.*

[16] S. Suri, "Minimum link paths in polygons and related problems," Ph.D. Dissertation, Department of Computer Science, Johns Hopkins University, 1987.

[17] R. E. Tarjan and C. J. Van Wyk, "An $O(n \log\log n)$-time algorithm for triangulating a simple polygon," *SIAM J. Computing 17*, No. 1, February 1988, pp. 143–178.

Efficient Geometric Algorithms in the EREW-PRAM

Danny Z. Chen*

Department of Computer Science
Purdue University
West Lafayette, IN 47907.

Abstract

We present a technique that can be used to obtain efficient parallel algorithms in the EREW-PRAM computational model. This technique enables us to optimally solve a number of geometric problems in $O(\log n)$ time using $O(n/\log n)$ EREW-PRAM processors, where n is the input size. These problems include: computing the convex hull of a sorted point set in the plane, computing the convex hull of a simple polygon, finding the kernel of a simple polygon, triangulating a sorted point set in the plane, triangulating monotone polygons and star-shaped polygons, computing the all dominating neighbors, etc. PRAM algorithms for these problems were previously known to be optimal (i.e., $O(\log n)$ time and $O(n/\log n)$ processors) only in the CREW-PRAM, which is a stronger model than the EREW-PRAM.

1 Introduction

We present a technique for obtaining efficient parallel algorithms in the EREW-PRAM computational model. This technique, when applied to a number of geometric problems on sorted point sets and simple polygons, enables us to get optimal solutions (in $O(\log n)$ time using $O(n/\log n)$ EREW-PRAM processors). The technique consists of a binary tree data structure (to be defined in Section 3), several kinds of parallel operations on the binary tree, and a combination of a two-way divide-and-conquer and a quarter-root divide-and-conquer strategies that are used in conjunction with the parallel tree operations. The parallel tree operations include parallel searching, parallel concatenation, and parallel split. These parallel tree operations are performed without read conflicts. This technique is likely to be useful for obtaining efficient parallel algorithms in the EREW-PRAM for problems other than those discussed in this paper.

The computational model we use is the EREW-PRAM (Exclusive Read Exclusive Write Parallel Random Access Machine); it is a synchronous parallel computational model in which all processors share a common memory and each processor can access any memory location in constant time. The EREW-PRAM does not allow more than one processor to simultaneously access the same memory address. We also refer to another version of the PRAM model, called the CREW (Concurrent Read Exclusive Write) PRAM. The CREW-PRAM allows simultaneous accesses to the same memory location by multiple processors *if* all such

---

*This research was partially supported by the Office of Naval Research under Grants N00014-84-K-0502 and N00014-86-K-0689, the National Science Foundation under Grant DCR-8451393, and the National Library of Medicine under Grant R01-LM05118.

concurrent accesses are for reading data only. The CREW-PRAM is obviously a more powerful model than the EREW-PRAM, and the simulation of a CREW-PRAM algorithm on an EREW-PRAM, using the same number of processors, can cost an increase in the time complexity by a logarithmic factor.

We use the technique to optimally solve the following problems: computing the convex hull of n sorted points in the plane (and hence the dual problem of finding the common intersection of n half-planes given sorted by their slopes), computing the convex hull of an n-vertex simple polygon, finding the kernel of an n-vertex simple polygon, triangulating n sorted points in the plane, triangulating an n-vertex monotone polygon or star-shaped polygon, and computing the *all dominating neighbors* of n values. Our EREW-PRAM algorithms for these problems all take $O(\log n)$ time using $O(n/\log n)$ processors.

The problems of computing the convex hulls of point sets and polygons, computing the kernel of a simple polygon, and triangulating point sets and polygons, are of fundamental importance in computational geometry and have applications in many areas. A great deal of work, both in the sequential and parallel computational models, has been done on finding efficient solutions for these problems (see [22, 27] for the sequential algorithms for these problems). For the problem of computing the convex hull of n *arbitrary* points in the plane, optimal solutions (i.e., $O(\log n)$ time and $O(n)$ processors) have been given in the CREW-PRAM [1, 4, 5] and in the EREW-PRAM [26]. Optimal CREW-PRAM algorithms were also known for the problem of triangulating n *arbitrary* points in the plane [25, 32] and for the problem of triangulating polygons [13, 33].

The problems we consider in this paper all have an obvious lower bound of linear work, and sequential linear time algorithms for them have already been known. Some of the solutions can be found in [6, 11, 14, 15, 16, 18, 21, 23, 30, 31]. (Our algorithms are optimal in the EREW-PRAM since they all run in $O(\log n)$ time and their *time* × *processors* products match the lower bound of these problems.)

Efficient CREW-PRAM algorithms solving the problems that we consider in this paper have also been discovered. The convex hull problem for a sorted point set can be solved in $O(\log n)$ time using $O(n/\log n)$ CREW-PRAM processors [6, 12, 29], and such an algorithm implies (by duality) the same complexity bounds for computing the common intersection of n half-planes whose slopes are given sorted [12]. Note that in the case where the points are already given sorted, the EREW-PRAM algorithm of [26] still requires $O(n)$ processors, which is sub-optimal. Our algorithm for the convex hull of a sorted point set can be viewed as another optimal algorithm (i.e., $O(\log n)$ time and $O(n)$ processors) in the EREW-PRAM for the case of unsorted input, because we can first obtain a sorted point set with $O(n)$ processors [8] and then use $O(n/\log n)$ processors for the remaining computation. For the case where the input points are given as a list of vertices on a simple polygon, the convex hull problem can be solved optimally in $O(\log n)$ time using $O(n/\log n)$ CREW-PRAM processors [29].

The problem of computing the kernel of a simple polygon has been solved optimally in the CREW-PRAM by Cole and Goodrich [9]. Their algorithm is based on the interesting observations which characterize the "curvature" of the polygon boundary.

For sorted point sets and monotone polygons, the triangulation problems can be solved optimally in the CREW-PRAM [6, 13, 18, 30]. In fact, Goodrich [13] showed that if the trapezoidal decomposition of a polygon (possibly with holes) has been provided, then a triangulation for that polygon can be done in $O(\log n)$ time using $O(n/\log n)$ CREW-PRAM processors. A CREW-PRAM algorithm in $O(\log n)$ time with $O(n/\log n)$ processors for triangulating a star-shaped polygon is recently given in [24].

The problem of computing the *all dominating neighbors* of n values is defined as follows: Given values $w_1, w_2, \ldots, w_n$, find for each index i the largest (resp., smallest) index $j < i$ (resp., $k > i$) such that $w_j \geq w_i$ (resp., $w_k \geq w_i$). This problem was considered by [6, 18] to be fundamentally important for solving several other problems (not only in computational geometry) in the PRAM. Especially, this problem was used as the basic subproblem for triangulating point sets in the plane and monotone polygons [6, 18, 25, 30]. Optimal algorithms for this problem (in $O(\log n)$ time and $O(n/\log n)$ CREW-PRAM processors) have been given in [6, 18, 30].

Most of the constituent parts of our algorithms, namely, the geometric observations, the divide and conquer strategies, the binary tree data structure, and the parallel tree operations (except for the parallel split), have been used before (for example, see [3, 5, 9, 12, 13, 28]). Our contribution is in putting these already available "parts" together in such a way that will enable us to avoid read conflicts that occurred in the previous known CREW-PRAM algorithms.

The rest of this paper is organized as follows. Section 2 gives the notation we use in the paper and outlines the general structure of the algorithms. Section 3 discusses the binary tree data structure and the parallel tree operations. Sections 4 to 6 show how to solve the problems we mentioned above.

2 Notation and Basic Algorithm Structure

Let S be a set of n points $p_1, p_2, \ldots, p_n$. S is sorted either by x-coordinate, or by polar angle with respect to a specified *polar point* $q \in S$. Let P be a simple polygon defined by the list of vertices $v_1, v_2, \ldots, v_n$, in the order of a clockwise travel along the polygon boundary.

WLOG (without loss of generality), we assume that in S (resp., P), no two points (resp., vertices) have the same x- or y-coordinate and no three points (resp., vertices) are collinear. The general situations can be taken care of by slightly modifying our algorithms.

We say two point sets S' and S'' are *separable* if there exists a vertical line such that S' and S'' are on the opposite sides of the line. Furthermore, for $k \geq 2$, we say k point sets are *separable* if there exist $k-1$ vertical lines such that for any two point sets, at least one of the $k-1$ vertical lines separates them.

The main procedure of our algorithms has a basic structure, which is the same as the one used in [3]. We outline it as follows.

Input: A set X of size m, which is either a sorted point set or the vertex set of a simple polygonal chain, and a positive integer d.

Output: The desired output $T(X)$ represented by a tree data structure.

Case 1. If $m \leq d$, then compute $T(X)$ with one processor in $O(m)$ time, using a sequential linear-time algorithm.

Case 2. If $d < m \leq d^2$, then divide X into two subsets X_1 and X_2 of equal size and recursively solve the two subproblems in parallel. Then compute $T(X)$ from $T(X_1)$ and $T(X_2)$, in $O(\log m)$ time using one processor.

Case 3. If $m > d^2$, then partition X into $g = (m/d)^{1/4}$ subsets $X_1, X_2, \ldots, X_g$ of size $m^{3/4}d^{1/4}$ each. Then, in parallel, recursively solve the g subproblems. Finally, compute $T(X)$ from $T(X_1), T(X_2), \ldots, T(X_g)$, in $O(\log m)$ time using $m/d = g^4$ processors.

end.

Observe that, if we could perform the various cases of the above outline within the claimed bounds, then the algorithm would run in $O(d + \log m)$ time with $O(m/d)$ processors since the recurrences of the time and processor complexities are (almost) the same as those in [3] (the only difference is that Case 2 in [3] runs in $O(\log^2 m)$ time while it takes only $O(\log m)$ time here). Choosing $d = \log m$, the above implies a time bound of $O(\log m)$ and a processor bound of $O(m/\log m)$. Therefore, a call to the algorithm with input $(X, \log n)$, $|X| = n$, will compute $T(X)$ in $O(\log n)$ time using $O(n/\log n)$ processors. The rest of the paper shows how to solve the problems by using algorithms like the one outlined above.

3 The Binary Tree Data Structure and Operations

The algorithms make use of a binary tree data structure. The definition of this tree structure is similar to the one for *hull tree*, used by Goodrich to store the information of the convex hull for a set of points [12] or the monotone funnel polygons [13]. We call such trees the *rank trees* because they support efficient operations based on the ranks of the leaves in the trees. A rank tree T is a binary search tree with a set of points stored at its leaves in some specified order (e.g., by increasing x-coordinate or polar angle). WLOG, we assume that the points are sorted by increasing x-coordinate. The leaves of T are doubly linked together. We denote the height of T (the length of the longest root-to-leaf path in T) by $h(T)$. Let T_v be the subtree of T rooted at node v of T. Each internal node v of T has four labels: the first stores the number of leaves in T_v, the second stores the point p at a leaf of T_v such that p has the smallest x-coordinate among the points stored at the leaves of T_v, and the other two respectively store the *predecessor* and the *successor* of p in the sorted point set stored at the leaves of T. In $O(h(T))$ time, one processor can search in T for an x-coordinate (and hence for a point) or search for the i-th ranked point stored in T (i.e., the i-th leaf of T in the left-to-right order) using the first or the second label stored in the internal nodes, respectively.

In the rest of this paper, all trees are assumed to be rank trees unless otherwise specified.

Our algorithms may need many processors to simultaneously search in such a tree. The next lemma, which is based on the parallel searching scheme of [28], enables us to avoid read conflicts during such parallel searching.

Lemma 1 ([3]) *Given a tree T, suppose each of k processors wants to perform a search in T. Two types of searches are allowed: the first type is a search for a particular point using its x-coordinate, and the second is of the type "find the t-th leaf of T starting from leaf l and moving to the right". Then the k processors can perform their searches in $O(\log k + h(T))$ time, without any read conflict.*

Proof. Same as that for Corollary 5.2 of [3] and hence omitted. □

The next two lemmas are for the parallel concatenation and split.

Lemma 2 (Goodrich [12, 13]) *Let S_1, S_2, ..., S_k be subsets of a point set S' separated by $k-1$ vertical lines, and let the trees $T(S_1)$, $T(S_2)$, ..., $T(S_k)$ for the subsets be given. Then tree $T(S')$ for S' can be built in $O(\log k + h)$ time using k EREW-PRAM processors, where h is the maximum of the $h(T(S_i))$'s. Also, $h(T(S')) = O(h + \log k)$.*

Proof. Same as the proof of Lemma 4.3 in [13] and hence omitted. □

Lemma 3 *Given a tree T and a list $(x_1, x_2, \ldots, x_k)$ of x-coordinates, suppose k EREW-PRAM processors want to split T into $k+1$ trees $T_0, T_1, \ldots, T_k$ such that all points in T_i have their x-coordinates within interval $[x_i, x_{i+1}]$, for $i \in \{0, 1, \ldots, k\}$ ($x_0 = -\infty$ and $x_{k+1} = +\infty$). Then the parallel split can be done in $O(\log k + h(T))$ time.*

Proof. WLOG, we assume that the list $(x_1, x_2, \ldots, x_k)$ is given sorted (otherwise, we have enough processors to do the sorting in $O(\log k)$ time [8]). Let processor P_i have value x_i, $i = 1, 2, \ldots, k$. Each P_i does the split operation in the same way as shown in Lemma 3.2 of [12]. That is, it searches for x_i in a tree and, when going down the tree following a root-to-leaf path, it makes two copies of the root-to-leaf path, whose nodes have the appropriately modified data from the original root-to-leaf path (actually, the original path is replaced by one copy); after it reaches the leaf of the path, it retraces the two paths which it just created, to update the labels of the nodes on the paths (see Lemma 3.2 in [12] for more details).

To avoid read conflicts, we use a procedure consisting of $O(\log k)$ stages. Let $T_{1,k} = T$. Before a tree $T_{a,b}$ is split ($a \leq b$), there is a group of processors $P_a, P_{a+1}, \ldots, P_b$ associated with it. The following is the general step. Suppose (the root of) $T_{a,b}$ is available in stage i but not in stage $i-1$. If $a = b$, then P_a splits $T_{a,b}$ at its root; otherwise, $P_{\lceil (a+b)/2 \rceil}$ splits $T_{a,b}$ at its root (by making two copies) and makes the roots of $T_{a,\lceil (a+b)/2 \rceil - 1}$ and $T_{\lceil (a+b)/2 \rceil + 1, b}$ available for stage $i+1$ (no split is done on $T_{\lceil (a+b)/2 \rceil + 1, b}$ if $\lceil (a+b)/2 \rceil + 1 > b$). The processors stop when they reach the leaves on the root-to-leaf paths they follow. After all processors stop at the leaves of the $k+1$ trees so obtained, we let each processor retrace the leaf-to-root paths in each tree which it just created and update the labels of the nodes on the paths just as was done in [12].

The correctness of this parallel split procedure is guaranteed by the facts that $(x_1, x_2, \ldots, x_k)$ is sorted and the split is based on searching the x_i's. No read conflict can occur in the procedure because although in the searching, different processors may follow the same root-to-leaf path in T, the processors, when doing the split, actually use different copies of the path, and such copies would have been created in the previous stages of the procedure. The time complexity of the procedure is clearly $O(\log k + h(T))$. □

4 Computing the Convex Hull of Sorted Points

This section discusses the algorithm for computing the convex hull of a point set in the plane sorted by increasing x-coordinate. Let $S = \{p_1, p_2, \ldots, p_n\}$ be a set of sorted points. We denote the convex hull of S by $CH(S)$. Points p_1 and p_n are both vertices of $CH(S)$ because p_1 and p_n, respectively, have the smallest and the largest x-coordinates among the points in S. Traveling along $CH(S)$ from p_1 to p_n clockwise, the portion of $CH(S)$ so visited is called the *upper hull* of S, denoted by $UH(S)$. Similarly, the portion of $CH(S)$ visited by traveling along $CH(S)$ from p_n to p_1 clockwise is called the *lower hull* of S, denoted by $LH(S)$. Due to the similarity in the computation of $UH(S)$ and $LH(S)$, we only discuss the algorithm for $UH(S)$. For two upper hulls $UH(S')$ and $UH(S'')$, where S' and S'' are separable point sets, the *upper common tangent* between $UH(S')$ and $UH(S'')$ is the common tangent of $UH(S')$ and $UH(S'')$ such that both $UH(S')$ and $UH(S'')$ are below it. The *lower common tangent* for two lower hulls is defined similarly. In the rest of this section, we just say the "common tangent" to mean the "upper common tangent".

The following two known results are useful.

Lemma 4 (Goodrich [12, 13]) *Let two upper hulls $UH(S_1)$ and $UH(S_2)$ be stored in trees T_1 and T_2, respectively, where S_1 and S_2 are two separable point sets. Then in $O(h(T_1)+h(T_2))$ time, one processor can find the common tangent between $UH(S_1)$ and $UH(S_2)$.*

Proof. See Lemma 3.1 in [12]. □

Lemma 5 (Atallah, Goodrich [5]) *Let S_1 and S_2 be two separable point sets with both $|S_1|$ and $|S_2|$ being $O(m)$, and let $UH(S_1)$ and $UH(S_2)$ be their upper hulls stored in two arrays, respectively. Then the common tangent between $UH(S_1)$ and $UH(S_2)$ can be computed in $O(c^2) = O(1)$ time using $m^{1/c}$ CREW-PRAM processors, where c is a positive constant.*

Proof. See Theorem 1 and **Algorithm A** in [5]. □

We immediately have the following corollary.

Corollary 1 *Let S_1 and S_2 be two separable point sets with both $|S_1|$ and $|S_2|$ being $O(m)$, and let $T(S_1)$ and $T(S_2)$ be two trees storing the upper hulls $UH(S_1)$ and $UH(S_2)$, respectively. Then the common tangent between $UH(S_1)$ and $UH(S_2)$ can be computed in $O(\log m + h)$ time using $m^{1/c}$ EREW-PRAM processors, where c is a positive constant and h is the maximum of $h(T(S_1))$ and $h(T(S_2))$.*

Proof. Recall that **Algorithm A** in [5] partitions the two arrays for the two upper hulls $UH(S_1)$ and $UH(S_2)$ into subarrays, then it finds in which subarrays the common tangent lies, and it then recursively solves the problem in the (two) subarrays so found. We simulate **Algorithm A** using $m^{1/c}$ EREW-PRAM processors. Note that **Algorithm A** is in the CREW-PRAM and it requires $O(1)$ time (given c a constant). Since now $UH(S_1)$ and $UH(S_2)$ are stored in trees (instead of arrays), each access to a leaf of a tree requires $O(h)$ time and one processor. Parallel searching for the points at the leaves of a tree (without read conflicts) can be done in $O(\log m + h)$ time using the available processors by Lemma 1. Each time an array (or a subarray) is partitioned in **Algorithm A**, we can achieve the same effect by partitioning the leaves of the relevant tree by using the ranks of the leaves. Such a partition can also be done in $O(\log m + h)$ time by doing parallel searching in the tree by Lemma 1. The other steps of **Algorithm A** can be easily simulated in $O(\log m)$ time in the EREW-PRAM. □

Now we show the algorithm for computing $UH(S)$. We refer to the cases of the outline in Section 2. In Case 1, we call the linear time algorithm in [14] to compute the upper hull. In Case 2, we use Lemma 4 to compute the common tangent between the two upper hulls returned from the two recursive calls, then we split the tree for each upper hull to remove the portion of that upper hull (if any) that is under the common tangent. The portions of the two upper hulls that remain form the upper hull that we seek in this case (by doing a simple concatenation). Since we use only one processor in this case, no read conflict occurs. We perform Case 3 as follows. Given g subsets $S_1, S_2, \ldots, S_g$ of a point set S', separated by $g-1$ vertical lines, where $|S'| = m$ and $g = (m/d)^{1/4}$, and given the trees $T(S_1), T(S_2), \ldots, T(S_g)$ representing their upper hulls, respectively, we compute the common tangent C_{ij} for each pair of $UH(S_i)$ and $UH(S_j)$, $1 \le i < j \le g$. Recall that we have $g^4 = m/d$ processors to do so, and $|S_k| = m^{3/4}d^{1/4}$ for each k. Every C_{ij} is obtained in $O(\log m)$ time using $g^2 = (m/d)^{1/2}$ processors by Corollary 1 (it has been shown in [3] that $h(T(S_k))$ is $O(\log m)$ for every k). Note that, in this case, we do not use the procedure for Lemma 4 to compute the C_{ij}'s. This is because the procedure for Lemma 4 searches for the points, on two upper hulls, where the common tangent for the two upper hulls lies, and before the search starts, we do not know at all which points we are getting to. In Case 3, each $T(S_k)$ is involved in the computation of $O(g)$ common tangents. If we used the procedure for Lemma 4, we would have read conflicts from the $O(g)$ simultaneous searches in $T(S_k)$, since we could not prearrange the $O(g)$ processors doing the searches (as was done in the scheme of [28]) in order to avoid read conflicts. From the $O(g^2)$ common tangents (the C_{ij}'s), we can find the portions of the $UH(S_k)$'s that form $UH(S')$, by doing parallel prefix [19, 20] (see [12] for the details on how this is done). The tree $T(S')$ is then built using Lemma 2.

5 Triangulating a Trapezoidally Decomposed Polygon

This section deals with the algorithm for triangulating an n-vertex polygon P (possibly with holes) when given a trapezoidal decomposition of P. Goodrich [13] showed how to triangulate a polygon in $O(\log n)$ time using $O(n/\log n)$ CREW-PRAM processors, provided that the trapezoidal decomposition of the polygon has been given (the trapezoidal decomposition of P can be done in $O(\log n)$ time using $O(n)$ CREW-PRAM processors [2]). Here we assume that the same input as in [13] is given. We will basically perform the same computation as in [13] (hence the reader is referred to [13] for more details of the algorithm). We only show how to use a *quarter-root* divide and conquer strategy and the parallel tree operations to perform various operations of [13] in the required time and processor complexities without read conflicts.

There are three phases in [13]. The reader is referred to [13] for the definitions used here. There is no read conflict in **Phase One**, whose goal is to construct the set of *one-sided monotone polygons* which decomposes P (by using parallel prefix [19, 20] and list ranking [10]). There is also no read conflict in **Phase Three**, whose goal is to triangulate the set of *monotone funnel polygons* resulting from **Phase Two** (by using parallel prefix and parallel merging [7, 17]). Hence we only need to concern ourselves with **Phase Two**, whose goal is to decompose every one-sided monotone polygon (from **Phase One**) into a set of monotone funnel polygons.

The difficult computation in **Phase Two** is to decompose every one-sided monotone polygon whose size is larger than $\log n$ into a set of monotone funnel polygons. Given a monotone chain C (from the one-sided monotone polygon), $|C| = m$, the procedure in **Phase Three** for this computation first partitions chain C into α subchains of equal size, and recursively solves the α subproblems in parallel. Then, from the results for the α subproblems, it computes the *bases* of the monotone funnel polygons that consist of the decomposition of the one-sided monotone polygon. Finally, it computes the left and right boundaries of the monotone funnel polygons and the lower hull of C. This procedure, although being quite complicated, essentially consists of the following operations: parallel prefix, sorting $O(\alpha)$ values, computing $O(\alpha^2)$ lower common tangents (among the lower hulls of the α subchains represented by α trees, as returned by the recursive calls), parallel splits on the α trees (into $O(\alpha)$ trees), and parallel concatenations of $O(\alpha)$ trees (to construct new trees representing the left and right boundaries of the monotone funnel polygons as well as the lower hull of chain C). Recall that our algorithm is based on the outline given in Section 2. Case 1 and Case 2 of this algorithm can be easily handled in the required complexity bounds. In Case 3, we have $g = (m/d)^{1/4}$ subproblems and g^4 processors. The parallel prefix and sorting [8] can be done using the available number of processors. The $O(g^2)$ lower common tangents are computed in a way similar to the convex hull algorithm of Section 4 (i.e., by Corollary 1). The parallel splits are done by Lemma 3 and the parallel concatenations are done by Lemma

2. None of these operations introduces read conflicts and all of them can be performed in $O(\log m)$ time using g^4 EREW-PRAM processors.

6 Other Geometric Algorithms

The algorithms described in Sections 4 and 5 enable us to obtain optimal EREW-PRAM algorithms for other geometric problems. All algorithms in this section take $O(\log n)$ time using $(n/\log n)$ EREW-PRAM processors.

If a point set in the plane is given sorted by polar angle with respect to a polar point q, the convex hull algorithm in Section 4 can be slightly modified to compute the convex hull of this point set. Using the convex hull algorithm in Section 4, we compute the convex hull of an n-vertex simple polygon P as follows: apply the visibility algorithm by Atallah, Chen, and Wagener [3] (which runs in $O(\log n)$ time using $O(n/\log n)$ EREW-PRAM processors) to P to obtain a point set sorted by x-coordinate (the vertices of P that are visible from the point $q = (0, +\infty)$), then apply the convex hull algorithm in Section 4 to the visible vertex set to find the upper hull of P. Using the geometric duality transformation [27], an immediate result from the convex hull algorithm in Section 4 is an optimal EREW-PRAM algorithm for the problem of computing the common intersection of n half-planes given sorted by their slopes. The kernel of a simple polygon can be computed optimally by using the convex hull algorithm in Section 4 as a subroutine in the algorithm of [9] (parallel prefix and parallel merging [7, 17] are also used in [9]).

The triangulation algorithm in Section 5 implies an optimal EREW-PRAM solution for triangulating a monotone polygon P, since a parallel merging will decompose P into a set of one-sided monotone polygons (triangulating one-sided monotone polygons is done in Section 5). Using the algorithm for computing the kernel of a simple polygon, we can check whether a simple polygon P is star-shaped or not. If it is, then the kernel of P is nonempty. Let r be the ray starting from vertex v_1 of P and going through a point q in the kernel of P. WLOG, we assume that r does not contain any edge of P. Let r intersect the boundary of P at a point $p \neq v_1$. If p is at some vertex of P, then we partition the boundary of P into two polygonal chains C' and C'' by r. Otherwise, let p be on edge e with endpoints v_i and v_{i+1}, and let C' be the polygonal chain consisting of vertices $v_1, v_2, \ldots, v_i$, and C'' the polygonal chain consisting of $v_{i+1}, v_{i+2}, \ldots, v_n, v_1$. Clearly, C' and C'' are both star-shaped (i.e., they are all visible from q). A triangulation for each of C' and C'' can be done in a way similar to the algorithm in Section 5 for triangulating a one-sided monotone polygon, since the vertices of P are sorted along the boundary of P in polar angle with respect to point q (in this case, q plays the role for C' and C'' same as the distinguished edge of a one-sided monotone polygon does for the polygon). The triangulation of C' and C'' also gives a monotone funnel polygon inside P (with base e, the right boundary from the triangulation of C', and the left boundary from the triangulation of C''). This monotone funnel polygon is the only portion of P that has not yet been triangulated. A triangulation of P can be completed by doing a parallel merging (see **Phase Three** in [13]). Hence we optimally solve the problem of triangulating a star-shaped polygon.

Triangulating a point set in the plane sorted by x-coordinate can be reduced to that of triangulating a set of one-sided monotone polygons, as follows. We first construct a monotone chain with the sorted points being the vertices of the chain. Then we compute the convex hull of the monotone chain (by using the convex hull algorithm of Section 4). The convex hull and the monotone chain, together, partition the region bounded by the convex hull into a set of one-sided monotone polygons. Triangulating a point set in the plane sorted by polar angle with respect to a polar point q can be reduced to that of triangulating the interior and the exterior of a star-shaped polygon, which can be done optimally in a way similar to the triangulation algorithm for a star-shaped polygon.

Given n values w_1, w_2, ..., w_n, the *all dominating neighbors problem*, in fact, can be viewed as a rectilinear version of triangulating a monotone polygonal chain. That is, we reduce the n values into n points $(1, w_1)$, $(2, w_2)$, ..., (n, w_n), and let a rectilinear monotone polygonal chain C have the n points as part of its vertices. The chain C consists of only vertical and horizontal line segments. All the "common tangents" we compute, and all the "diagonals" we add to the "triangulation", are horizontal line segments. Our algorithm in Section 5 can be modified to solve this problem.

Acknowledgement. The author is very grateful to Professor Mikhail Atallah for his great support and valuable suggestions to this work.

References

[1] A. Aggarwal, B. Chazelle, L. Guibas, C. O'Dunlang and C. Yap. "Parallel computational geometry." *Algorithmica* 3 (3) (1988), 293–327.

[2] M. Atallah, R. Cole and M. Goodrich. "Cascading divide-and-conquer: a technique for designing parallel algorithm." *SIAM J. Comput.* 18 (3) (1989), 499–532.

[3] M. J. Atallah, D. Z. Chen and H. Wagener. "An optimal parallel algorithm for the visibility of a simple polygon from a point." TR 88-759, Dept. of Comput. Sci., Purdue Univ., April 1988. To appear in *Journal of the ACM.* A preliminary version appeared in *Proc. 5th Annual ACM Symp. on Computational Geometry*, 1989, pp. 114–123.

[4] M. J. Atallah and M. T. Goodrich. "Efficient parallel solutions to some geometric problems." *J. Parallel & Distributed Comput.* 3 (1986), 492–507.

[5] M. J. Atallah and M. T. Goodrich. "Parallel algorithms for some functions of two convex polygons." *Algorithmica* 3 (1988), 535–548.

[6] O. Berkman, D. Breslauer, Z. Galil, B. Schieber and U. Vishkin. "Highly parallelizable problems (Extended abstract)." *Proc. 21st Annual ACM Symp. on Theory of Computing*, 1989, pp. 309–319.

[7] G. Bilardi and A. Nicolau. "Adaptive bitonic sorting: An optimal parallel algorithm for shared-memory machines." *SIAM J. Comput.* 18 (1989), 216–228.

[8] R. Cole. "Parallel merge sort." *SIAM J. Comput.* 17 (4) (1988), 770–785.

[9] R. Cole and M. T. Goodrich. "Optimal parallel algorithms for polygon and point-set problems." *Proc. 4th Annual ACM Symp. Computational Geometry*, 1988, pp. 211–220. (To appear in *Algorithmica.*)

[10] R. Cole and U. Vishkin. "Approximate parallel scheduling. Part I: the basic technique with applications to optimal parallel list ranking in logarithmic time." *SIAM J. Comput.* 17 (1) (1988), 128–142.

[11] A. Fournier and D. Y. Montuno. "Triangulating simple polygons and equivalent problems." *ACM Trans. Graphics* 3 (2) (1984), 153–174.

[12] M. T. Goodrich. "Finding the convex hull of a sorted point set in parallel." *Inform. Process. Lett.* 26 (1987/88), 173–179.

[13] M. T. Goodrich. "Triangulating a polygon in parallel." *J. of Algorithms* 10 (1989), 327–351.

[14] R. L. Graham. "An efficient algorithm for determining the convex hull of a finite planar set." *Inform. Process. Lett.* 1 (1972), 132–133.

[15] R. L. Graham and F. F. Yao. "Finding the convex hull of a simple polygon." *J. Algorithms* 4 (4) (1983), 324–331.

[16] M. R. Garey, D. S. Johnson, F. P. Preparata and R. E. Tarjan. "Triangulating a simple polygon." *Inform. Process. Lett.* 7 (4) (1978), 175–179.

[17] T. Hagerup, and C. Rub. "Optimal merging and sorting on the EREW PRAM." *Inform. Process. Lett.* 33 (1989), 181–185.

[18] D. G. Kirkpatrick and T. Przytycka. "An optimal parallel algorithm for all dominating neighbors problem and its applications." Manuscript.

[19] C. P. Kruskal, L. Rudolph and M. Snir. "The power of parallel prefix." *IEEE Trans. Comput.* C-34 (1985), 965–968.

[20] R. E. Ladner and M. J. Fischer. "Parallel prefix computation." *J. of the ACM* 27 (4) (1980), 831–838.

[21] D. T. Lee. "On finding the convex hull of a simple polygon." *Int'l J. Comput. and Inform. Sci.* 12 (2) (1983), 87–98.

[22] D. T. Lee and F. P. Preparata. "Computational geometry — A survey." *IEEE Trans. Comput.* C-33 (12) (1984), 1072–1101.

[23] D. T. Lee and F. P. Preparata. "An optimal algorithm for finding the kernel of a polygon." *Journal of the ACM* 26 (1979), 415–421.

[24] C. J. Lee and W. J. Hsu. "Triangulating a star-shaped polygon in parallel." To appear in *Proc. 28th Annual Allerton Conf. on Communication, Control, and Computing*, Monticello, Illinois, 1990.

[25] E. Merks. "An optimal parallel algorithm for triangulating a set of points in the plane." *International Journal of Parallel Programming* Vol. 15, No. 5 (1986), 399–411.

[26] R. Miller and Q. F. Stout. "Efficient parallel convex hull algorithms." *IEEE Trans. Comput.* C-37 (12) (1988), 1605–1618.

[27] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.

[28] W. Paul, U. Vishkin and H. Wagener. "Parallel dictionaries on 2-3 trees." *Proc. 10th Coll. on Autom., Lang., and Prog. (ICALP), LNCS 154*, Springer, Berlin, 1983, pp. 597–609.

[29] H. Wagener. "Parallel Computational Geometry Using Polygonal Order." Ph.D. thesis, Technical University of Berlin, FRG, 1985.

[30] H. Wagener. "Triangulating a monotone polygon in parallel." *Computational Geometry and Its Applications, Proc. International Workshop on Computational Geometry (CG'88)*, Wurzburg, FRG, 1988, pp. 136–147.

[31] T. C. Woo and S. Y. Shin. "A linear time algorithm for triangulating a point-visible polygon." *ACM Trans. Graphics* 4 (1) (1985), 60–70.

[32] C. A. Wang and Y. H. Tsin. "An $O(\log n)$ time parallel algorithm for triangulating a set of points in the plane." *Inform. Process. Lett.* 25 (1987), 55–60.

[33] C. K. Yap. "Parallel triangulation of a polygon in two calls to the trapezoidal map." *Algorithmica* 3 (1988), 279–288.

1-D Compaction in the Presence of Forbidden Regions

Susanne Hambrusch [*]
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Hung-Yi Tu[†]
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Abstract

In this paper we consider the one-dimensional compaction problem when the layout area contains forbidden regions and the layout components are allowed to move across these regions. Given n layout components and k forbidden regions, each of rectangular shape, we show how to solve this compaction problem in $O(\Delta)$ time with $O((n+k)\log k + \delta\log\delta)$ preprocessing, where Δ and δ are measures for the interaction between layout components and forbidden regions, $\Delta < n^2k$, $\delta \leq nk$. We also consider special cases of the forbidden regions. For example, when every forbidden region is of length h, where h is the height of the layout, the compaction problem can be solved in $O(\rho\log\left\lceil\frac{nk}{\rho}\right\rceil + \rho)$ time, with $O(\rho + n\log n + k)$ preprocessing, where ρ is the number of edges in transitive closure of visibility graph induced by the layout components, $\rho < n^2$.

1 Introduction

A one-dimensional (1-D) compacter takes as an input a VLSI layout and generates a layout of smaller area by sliding the layout components in one direction [1, 2, 4]. W.l.o.g., let it be the horizontal direction. Early compaction algorithms have not been used as widely as expected. One of the reasons given attributes it to the limitations inherent to these compaction systems; e.g., they could not handle additional constraints on where to place or not to place certain layout components [5]. In this paper we take a step towards incorporating additional constraints into the compaction process. We consider compaction when the layout area contains forbidden regions. The forbidden regions can represent, for example, pre-positioned layout components or holes in the layout area. The positions of the forbidden regions cannot be altered during the compaction process, but layout components are allowed to "slide over" the forbidden regions. We assume that both forbidden regions and layout components are of rectangular shape.

Given are n rectangles, $R_1,R_2,\ldots,R_n$, and k forbidden regions, $B_1,B_2,\ldots,B_k$, with the edges of the rectangles and forbidden regions parallel to the coordinate axes. A *configuration* of the layout assigns to every lower left corner of a rectangle a position of the layout area. A configuration is called *feasible* if it keeps the relative order of the rectangles in the horizontal direction and no two rectangles and no rectangle and forbidden region overlap. A feasible configuration of minimum area is called a *minimum configuration.*

---

[*] Research supported in part by ONR under contracts N00014-84-K-0502 and N00014-86-K-0689, and by NSF under Grant MIP-87-15652.

[†] Research supported in part by NSF under Grant MIP-87-15652 and ONR under contract N00014-84-K-0502.

Let h be the height of the layout. Since compaction is done in the horizontal direction, h is determined by the forbidden regions and the rectangles, and is not altered during compaction. We first consider the compaction problem when every forbidden region has height h. We refer to this problem as the *k-partition problem.* We develop characterizations of a minimum configuration that allow us to determine a minimum configuration of the k-partition problem in $O(\rho \log \lceil \frac{nk}{\rho} \rceil + \rho + n \log n + k)$ time, where ρ is the number of edges in transitive closure of visibility graph induced by the n rectangles, $\rho < n^2$. We then generalize the approach used for the k-partition problem to handle the general problem, the *forbidden region problem.* Let δ be the number of pairs (i, j) such that rectangle R_i could overlap with forbidden region B_j (if we slid R_i horizontally), $\delta \leq nk$. We again characterize a set of feasible configurations and show that a minimum configuration is among them. The number of feasible configurations considered is at most $n + 1$ for the k-partition problem and at most δ for the forbidden region problem. In both algorithms we generate the configurations in an order that allows us to update changes in the positions of the layout (and thus the width of the layout associated with each configuration) efficiently. The running time for the forbidden region problem is $O(\Delta)$ with $O((n + k) \log k + \delta \log \delta)$ preprocessing, where Δ is another measure for the interaction between the layout components and the forbidden regions, $\Delta < n^2 k$.

The k forbidden regions in the k-partition problem can be viewed as a position in the layout where a vertical cut can be made. In certain environments one may need to make k cuts, but does not have the positions of the cuts pre-determined. Rather, the vertical cuts should be made so that the maximum distance between two consecutive cuts is a minimum. For example, in a multi-layer environment minimizing the maximum space between two cuts corresponds to minimizing the volume of the 3-dimensional layout. This is the objective in the *minmax k-partition problem.* For the case when only three cuts are made we present an $O(n)$ time algorithm. For arbitrary k we present an $O(\rho + n \log n)$ time algorithm, where ρ equals the number of edges in the transitive closure of the visibility graph. In this algorithm we also identify a set of feasible configurations. However, it is now possible to employ a binary search technique for identifying a minimum configuration.

2 k-partition problem

In this section we present our algorithm for the k-partition problem. Assume that the width of every forbidden region is zero. Straightforward modifications to the algorithm can handle forbidden regions with arbitrary widths. For convenience we introduce two fictitious rectangles R_0 and R_{n+1} of height h and width zero, which are initially positioned to the left and to the right of the other rectangles and the forbidden regions, respectively. Figure 1(a) shows an initial configuration of a 7-partition problem for $n = 8$. Let $x(B_i)$ be the position of forbidden region B_i. Let S_i denote the area of width d_i, called the *slot*, between B_i and B_{i+1} for $1 \leq i \leq k - 1$. For $i = 0$, S_i is the area available to the left of B_1, and for $i = k$, S_i is the area to the right of B_k, respectively. In any configuration C, let $(x_C(R_i), y_C(R_i))$ be the position of the lower left corner of rectangle R_i. Since the width of any minimum configuration is at least $x(B_k) - x(B_1)$, we only consider feasible configurations in which R_0 is to the left of position $x(B_1)$ and R_{n+1} is to the right of position $x(B_k)$. The width of configuration C is then the distance between R_0 and R_{n+1}. A feasible configuration C is called *left-compressed* if for any other feasible configuration C' in which $x_{C'}(R_0) = x_C(R_0)$ we have $x_C(R_i) \leq x_{C'}(R_i)$ for $1 \leq i \leq n$. Intuitively, in a left-compressed configuration all rectangles are positioned as far to the left as possible. Figure 1(b) shows the left-compressed version of the configuration shown in Figure 1(a). It is easy to see that performing a left-compression on a configuration cannot increase its width.

Two rectangles R_i and R_j are *visible from* each other if one can draw a horizontal line segment connecting R_i and R_j without intersecting any other rectangles. The *visibility graph* induced by the rectangles is the directed graph $G = (V, E)$ in which each vertex corresponds to a rectangle and the edges reflect the visibility between the rectangles. An edge is directed from i to j if rectangle R_j is to the right of R_i and R_i and R_j are visible from each other. The vertex associated with R_i has a weight of w_i, where w_i is the width of R_i. The length of a path from R_i to R_j is the sum of the weights of the vertices on this path. The length of the longest path from R_0 to R_i in a visibility graph G is denoted by l_i.

Depending on the layout system used to generate the initial layout, the visibility information between rectangles may or may not be available [6]. If it is not available, it can easily be determined in $O(n \log n)$ time. Throughout the paper we assume that the visibility graph G is available. Furthermore, we assume that the forbidden regions have been sorted by increasing x-coordinates, and that the sequence of rectangles is arranged in a topological order induced by G. All these preprocessing steps can be accomplished in $O(n \log n + k \log k)$. We next state a property of a minimum configuration.

Property 2.1 *Let C be a left-compressed minimum configuration. Then, there exists a rectangle R_i, $0 \leq i \leq n$, such that $x_C(R_0) + l_i = x(B_1)$.*

There are at most $n + 1$ left-compressed feasible configurations satisfying Property 2.1. An immediate algorithm for the k-partition problem is to generate these $n + 1$ configurations and to determine the minimum one among them. This is the basis of our first algorithm. Let C_i be the left-compressed configuration in which the width of slot S_0 is l_i, $0 \leq i \leq n$. The width of configuration C_i can easily be determined as follows. First, we set $x_{C_i}(R_0) = x(B_1) - l_i$. We then process the rectangles in order $R_1, R_2, \ldots, R_{n+1}$. When processing rectangle R_j we determine the rectangle R_{max} which is a predecessor of R_j in G_r and whose value $x_{C_i}(R_{max}) + w_{max}$ is a maximum among all such values. Assume that R_{max} is in slot S_l. In order to determine the position of R_j, the following query is answered. Let d'_l be the width still available in slot S_l. Given slots $S_l, S_{l+1}, \ldots, S_{k-1}, S_k$ with widths $d'_l, d_{l+1}, \ldots, d_{k-1}, +\infty$, respectively, determine the smallest index a such that $d_a \geq w_j$. If $a = l$, we set $x(R_j)$ to $x_{C_i}(R_{max}) + w_i$; otherwise, we set $x(R_j)$ to $x(B_a)$. By using a balanced tree, this query can be answered in $O(\log k)$ time. Thus, configuration C_i can be generated in $O(n \log k)$ time and the minimum configuration can be determined in $O(n^2 \log k)$ time. The generation of the balanced tree costs $O(k)$ time.

We now show how to improve the running time to $O(\rho \log \lceil \frac{nk}{\rho} \rceil + \rho + k)$. Since $\rho = O(n^2)$, we not only make the running time input-sensitive, but also improve its worst-case performance. The following property characterizes how the positions of the rectangles change in consecutive configurations. It is this characterization that allows us to make the running time of the algorithm input-sensitive. Assume now that the rectangles are topologically ordered so that $l_0 \leq l_1 \leq \ldots \leq l_{n+1}$ and that the configurations are generated in the order $C_n, C_{n-1}, \ldots, C_1, C_0$. In configuration C_n every rectangle is assigned a position in slot S_0. Recall that C_i and C_{i-1} are the left-compressed configurations with $d_0 = l_i$ and $d_0 = l_{i-1}$, respectively.

Property 2.2 *Let R_p be a rectangle not reachable from R_i in G_r. If, in configuration C_i, rectangle R_p is to the right of B_1, then the position of R_p in C_{i-1} is as in configuration C_i (i.e., $x_{C_{i-1}}(R_p) = x_{C_i}(R_p)$). If, in configuration C_i, rectangle R_p is to the left of B_1, then, in configuration C_{i-1}, it is shifted $l_i - l_{i-1}$ to the right (i.e., $x_{C_{i-1}}(R_p) = x_{C_i}(R_p) + (l_i - l_{i-1})$).*

Property 2.2 implies the followings. If we generate the configurations in the order $C_n, C_{n-1}, \ldots, C_0$, the positions assigned to rectangles cannot decrease. When generating configuration C_{i-1} from C_{i-1}, the rectangles in slot S_0 are pushed $l_i - l_{i-1}$ positions to the right. This pushing leaves all rectangles, except R_i, to the left of B_1 in slot S_0. Rectangle R_i is pushed to across B_1. The rectangles not reachable from R_i and to the right of B_1 do not need new positions in C_{i-1} (since sliding R_i to the right has no effect on them). The rectangles reachable from R_i may need a new position. In order to efficiently determine these new positions we build a level-linked finger tree [3] on the slots. A finger tree allows fast searching in the vicinity of a finger. We first use the finger tree to locate the new position for R_i in C_{i-1}. Assume we are determining a new position for rectangle R_j in configuration C_{i-1}. Let R_j be assigned a position in slot S_m in C_i. Let R_{max} be defined for rectangle R_j as before. Assume R_{max} is in slot S_l. If $l < m$, then R_j remains in slot S_m. Hence, assume that $l \geq m$ and let d'_l be again the width available in slot S_l for R_j in configuration C_{i-1}. Assume $w_j > d'_l$ (i.e., S_l now is not big enough for R_j.) Assume R_j's new position is in slot S_a. Then, using the finger tree for the searching, this slot is determined in $O(1 + \log{(a - l)})$ time.

We point out that our algorithm does not explicitly update the positions of the rectangles to the left of B_1, since doing so would be too time-consuming. For each configuration we only determine its width. Once we have the index i resulting the left-compressed minimum configuration, we re-build configuration C_i in $O(n \log k)$ time by setting $x_{C_i}(R_0) = x(B_1) - l_i$ and left-compressing the n rectangles.

We now show that this algorithm achieves the claimed time bound. Let r_i be the number of rectangles that can reach R_i (we assume that a rectangle can reach itself.) Initially, rectangle R_i is to the left of B_1. At some point R_i moves across B_1 and is assigned a new position in a slot to the right of B_1. From this point on, every time a rectangle that can reach R_i moves across B_1, R_i may get re-positioned. Its new position is always to the right of its old position. Let $f_{i,j}$ be the number of slots rectangle R_i moves to the right when R_i is re-positioned for the j-th time, $1 \leq j \leq r_i$. By using the finger tree to determine the new slots, the total time needed to re-position R_i is

$$\sum_{j=1}^{r_i} (1 + \log f_{i,j}) \leq r_i \log \tfrac{k}{r_i} + r_i.$$

Let T_p be the total time needed to re-position all rectangles. Then,

$$\begin{aligned} T_p &\leq \sum_{i=0}^{n} \left(r_i \log \frac{k}{r_i} + r_i \right) \\ &\leq \sum_{i=0}^{n} r_i \log k - \sum_{i=0}^{n} r_i \log r_i + \sum_{i=0}^{n} r_i \\ &\leq \rho \log k - \sum_{i=0}^{n} r_i \log r_i + \rho, \end{aligned}$$

where $\sum_{i=0}^{n} r_i = \rho$. A straightforward computation shows that $\sum_{i=0}^{n} r_i \log r_i > \rho \log \frac{\rho}{n}$. Using this lower bound on $\sum_{i=0}^{n} r_i \log r_i$, we get $T_p < \rho \log \left\lceil \frac{nk}{\rho} \right\rceil + \rho$. We conclude this section with the following result.

Theorem 2.1 *Given n rectangles and k forbidden regions, the k-partition problem can be solved in $O(\rho \log \left\lceil \frac{nk}{\rho} \right\rceil + \rho)$ time with $O(k + n \log n + \rho)$ preprocessing time.*

3 Forbidden region problem

In this section we generalize the approach developed in the previous section to solve the forbidden region problem. We again use two fictitious rectangles R_0 and R_{n+1}, each being of width 0 and of height h. Rectangle R_0 is always positioned to the left and R_{n+1} always to the right of all forbidden regions, respectively. Our algorithm will only generate left-compressed configurations. Clearly, left-compressing a configuration cannot increase its width.

Every rectangle R_i has now a set of slots, $\mathcal{S}_i$, associated with it. Set $\mathcal{S}_i$ is determined as follows. We say forbidden region B_j and rectangle R_i are *related* if we can draw a horizontal line intersecting both B_j and R_i. Assume rectangle R_i and forbidden region B_j are related and let b_j be the width of B_j. Consider the rectangular region of maximal width that has position $(x(B_j) + b_j, y(R_i))$ as its lower left corner, has a height equal to the height of R_i and does not intersect any other forbidden region. If the width of this region is at least w_i (i.e., R_i can be placed into in), then this region represents a slot in set $\mathcal{S}_i$. We also include into set $\mathcal{S}_i$ two special slots of infinite width. Namely, the slot whose right border coincides with the left border of the leftmost forbidden region and the slot whose left border coincides with the right border of the rightmost forbidden region related to R_i. Our first property is a generalization of Property 2.1 of the k-partition problem.

Property 3.1 *Let C be a left-compressed minimum configuration. Then, there exists a rectangle R_i, $0 \leq i \leq n$, and a forbidden region B_j, $1 \leq j \leq k$, such that $x_C(R_0) + l_i = x(B_j)$ and $x_C(R_i) + w_i = x(B_i)$.*

Let q_i be the number of forbidden region rectangle R_i is related to (i.e., $q_i = |\mathcal{S}_i|$), and let $\delta = \sum_{i=0}^{n} q_i \leq kn$. Property 3.1 states that the minimum configuration is one among δ configurations. Property 3.1 can easily be proven by contradiction and its proof is omitted. Observe that the requirement that R_0 gets positioned to the left of all forbidden regions is necessary to make Property 3.1 true. It is possible that $i = 0$ is the only index in the minimum left-compressed configuration for which the property holds.

Our algorithm generates the δ configurations in an order that allows us to update the necessary information about new positions of the rectangles efficiently. The positions of rectangle R_0 in the δ configurations are determined in $O(\delta)$ time (by using the l_i's and the $x(B_j)$'s). We then order these positions of rectangle R_0 by increasing x-values. At this time we also discard any configurations in which $x(R_0)$ is greater than the x-position of the leftmost forbidden region (obviously, these configurations are not feasible). Let $C_1, C_2, \cdots, C_{\delta-1}, C_\delta$ be the left-compressed configurations with $x_{C_a}(R_0) < x_{C_{a+1}}(R_0)$. Not every one of these δ configurations represents necessarily a feasible configuration. Let C_a be a configuration in which the position of R_0 is dictated by R_i and B_j. If C_a is feasible, then every rectangle R_t on the longest path from R_0 to R_i is positioned at $x_{C_a}(R_0) + l_t - w_t$. We say that the path from R_0 to R_i is *tight.* Configuration C_a is not feasible when forbidden regions block the tight path from R_0 to R_i. The test of whether configuration C_a is indeed feasible is done during the algorithm.

We next describe how to generate the configurations. It is clear that, when generating the configurations in the order of increasing x-value of the position of R_0, a rectangle can only move to the right. Our algorithm organizes each set $\mathcal{S}_i$ as a linear list containing the slots with increasing x-positions. Each list $\mathcal{S}_i$ will be traversed at most once during the algorithm. The initial configuration C_1 is generated by positioning R_0 at the associated position and performing a left-compression. Assume now that we have decided whether configuration C_a is feasible and, if it is, have determined its width. While generating the configurations we maintain for every rectangle R_i a variable $x'(R_i)$ which contains partial information about the position of R_i in configuration C_a. The value of $x_{C_a}(R_i)$, the actual position of R_i in C_a, is $\max\{x'(R_i), x_{C_a}(R_0) + l_i - w_i\}$. In some sense, $x'(R_i)$ contains the correct position of R_i whenever the path from R_0 to R_i in C_a is not tight.

Let rectangle R_{i1} and forbidden region B_{j1} be the pair that dictates the position of R_0 in C_a (i.e., $x_{C_a}(R_0) = x(B_{j1}) - l_{i1}$). Let R_{i2} and B_{j2} be the pair that dictates the position of R_0 in C_{a+1}. Let $\epsilon = x_{C_{a+1}}(R_0) - x_{C_a}(R_0)$. We check in $O(1)$ time whether having rectangle R_{i2} at position $x(B_{j2}) - w_{i2}$ results in a feasible configuration as follows. We compute $x_{C_a}(R_{i2})$, the position of R_{i2} in configuration C_a. If $x_{C_a}(R_{i2}) + \epsilon = x(B_{j2}) - w_{i2}$, then the path from R_0 to R_{i2} in C_{a+1} is tight and C_{a+1} is feasible. We next describe how to compute the width of configuration C_{a+1}. The crucial insight into computing the width of C_{a+1} efficiently lies in the fact that, in order to compute the width, we only need to explicitly re-position the rectangles reachable from R_{i1}. When going from configuration C_a to configuration C_{a+1}, a rectangle R_p finds itself in one of four possible situations.

Case 1. Rectangle R_p is reachable from rectangle R_{i1}. Since R_{i1} moves from being immediately to the left of forbidden region B_{j1} to being immediately to the right of it, R_p may need a new position. Using the example shown in Figure 2, rectangle R_{p1} is reachable from R_{i1} and gets a new position assigned while R_{p2} keeps its position. We determine the new position by using set $\mathcal{S}_p$ and we record the new position in $x'(R_p)$.

Case 2. Rectangle R_p is not reachable from R_{i1} and the path from R_0 to R_p in configuration C_a is tight. Then, in configurations C_{a+1} rectangle R_p moves ϵ positions to the right. This situation applies to rectangle R_{p3} of Figure 2. The change in position is not explicitly recorded since doing so would be too time consuming. Recall that we are able to compute the position of any rectangle on a tight path on $O(1)$ time. It remains to be shown that moving rectangle R_p ϵ positions to the right always results in a feasible configuration (i.e., this does not cause R_p to overlap with any other rectangle or forbidden region). The proof of this fact is omitted due to space limitations.

Case 3. Rectangle R_p is not reachable from R_{i1}, the path from R_0 to R_p in C_a is not tight, and the path from R_0 to R_p in C_{a+1} is tight. The amount R_p moves to the right is now determined by ϵ minus the amount of "non-tightness" on the path from R_0 to R_p in configuration C_a. This situation applies to rectangle R_{p4} of Figure 2. Note that in configuration C_a $x'(R_p)$ contained the correct position of R_p, while in C_{a+1} the correct position is determined by $x_{C_{a+1}}(R_0) + l_p - w_p$. The argument that moving R_p to the right results in a feasible configuration is as given for Case 2.

Case 4. Rectangle R_p is not reachable from R_{i1} and the path from R_0 to R_p in neither tight in C_a nor C_{a+1}. In this situation rectangle R_p does not change its position when going from configuration C_a to configuration C_{a+1}. An example for this situation is rectangle R_{p5} in Figure 2.

We are now ready to give a complete description of our algorithm. The preprocessing step includes computing the x-position of rectangle R_0 in the δ configurations, arranging the configurations according to increasing x-value of R_0, and constructing the set $\mathcal{S}_i$ for every rectangle R_i, $1 \leq i \leq n$. These steps take $O(\delta \log \delta + (n+k)\log k)$ time. We then generate the configurations $C_1, \cdots, C_{\delta-1}, C_\delta$. When generating C_{a+1} from C_a we determine the new positions for rectangles reachable from R_{i1} as follows. Let R_p be a rectangle reachable from R_{i1} so that all predecessors of R_p for which Case 1 applies have been handled. Let R_m be the immediate predecessor of R_p for which Case 1 applies (there exists at least one) and for which $x_{C_{a+1}}(R_m) + w_m$ is a maximum. This index m is determined while the immediate predecessors of R_p are re-positioned. Rectangle R_p needs a new position if the condition

$$x_{C_{a+1}}(R_m) + w_m > max\{x'(R_p), x_{C_{a+1}}(R_0) + l_p - w_p\}$$

is satisfied. Note that the right-hand side of the condition does not correspond to the position of R_p in C_a. The quantity $x_{C_{a+1}}(R_0) + l_p - w_p$ already takes the shift to the right from position $x_{C_a}(R_0)$ to $x_{C_{a+1}}(R_0)$ into account. If the condition is true, rectangle R_p overlaps with another rectangle (not necessarily R_m). In order to determine R_p's new position, we locate, using $\mathcal{S}_p$, the leftmost position $\geq x_{Ca+1}(R_m) + w_m$. This position is located by a linear scan which starts at the slot containing the old position of R_p. When all rectangles reachable from R_{i1} have been handled, we compute the width of configuration C_{a+1}. This width is determined by $x_{C_{a+1}}(R(n+1)) - x_{C_{a+1}}(R(0))$ and is hence computed in $O(1)$ time. After the widths of all configurations have been computed, we re-build the left-compressed configuration giving minimum width in $O(\delta)$ time.

We now establish the claimed time bound and start with the time required for re-positioning the rectangles. Let r'_i be the number of rectangles which can be reached by rectangle R_i. Each time rectangle R_i is pushed across a forbidden region, we may have to re-position all the rectangles which can be reached by from R_i. Rectangle R_i is pushed across at most q_i forbidden regions. Hence, the total number of times R_i causes a re-positioning is at most $q_i r'_i$. Overall, we re-position at most $\Delta = \sum_{i=1}^{n} q_i r'_i$ rectangles. In order to find new positions for all rectangles the lists S_i, $1 \leq i \leq n$, are traversed. This costs an additional $O(\delta)$ time. Since $r'_i \geq 1$ (i.e., a rectangle can be reached by itself), we have $\Delta \geq \delta$. We can thus state the following theorem.

Theorem 3.1 *Given n rectangles and k forbidden regions, the forbidden region problem can be solved in $O(\Delta)$ time with $O((n+k)\log k + \delta \log \delta)$ preprocessing time.*

4 Minmax k-partition problem

When each of the k forbidden regions has height h, the forbidden regions model positions in the layout area where a vertical cut can be made. In certain environments one may need to make k cuts, but does not have the positions of the cuts pre-determined. Rather, the cuts should be made so that the maximum distance between two consecutive cuts is minimized. We refer to this problem as the minmax k-partition problem and present an algorithm to solve it in $O(\rho + n \log n)$ for arbitrary k. For the case when the layout components get separated by only one cut we present an $O(n)$ time algorithm.

We start by giving a more formal definition of the minmax k-partition problem. Given are again n rectangles, $R_1, R_2, \ldots, R_n$, where R_i has width w_i. We are to determine the position of k vertical cuts so that all rectangles are to the right of the first cut and to the left of the k-th cut, respectively, and no rectangle intersects a cut (i.e., the rectangles are partitioned into $k-1$ groups). Let $B_1, B_2, \ldots, B_k$ be the cuts, $S_1, S_2, \ldots, S_{k-1}$ be the slots (i.e., the area between two consecutive cuts), and let $d_1, d_2, \ldots, d_{k-1}$ be the distances between two consecutive cuts. Let $d^* = \max_{1 \leq i < k-1} d_i$. A minimum configuration for the minmax k-partition problem is one in which d^* is a minimum. We also refer to d^* as the *width* of the partition. Observe that the statement of the problem requires $k \geq 2$. For $k = 2$, it corresponds to the standard compaction problem and for $k = 3$ it corresponds to compacting the rectangles onto 2 layers.

The following property characterizes a relationship between the width of a minimum configuration and the length of the longest path between two rectangles. It is the basis for reducing the search space containing the minimum configuration.

Property 4.1 *Let C be a minimum configuration of width d^*. Then, there exist rectangles R_i and R_j and two consecutive cuts B_a and B_{a+1} such that $d^* = d_a = l_{i,j}$ where $l_{i,j}$ is the length of the longest path from R_i to R_j in G.*

This property states that the width of a minimum configuration is equal to the length of the longest path between two rectangles. Our algorithm first generates the necessary $l_{i,j}$'s. For every rectangle R_i we determine the length of the longest paths from R_i to all rectangles reachable from R_i. Using G, these values are generated in $O(\rho)$ time.

We then employ a binary search strategy to find d^*. Our algorithm uses as a procedure that, given a value d, determines the minimum number of cuts needed to achieve a partition of width d. Let MIN_CUT(d) be this procedure. Using G, MIN_CUT(d) generates the number of cuts needed in $O(n)$ time as follows. Assume we have an infinite number of cuts $B_1, B_2, \ldots$ with the distance between two consecutive cuts being d. We now process the rectangles in a topological order induced by G. Let R_i be the rectangle currently being processed, and R_{max} be the rectangle that is a predecessor of R_i in G and for which $x(R_{max}) + w_{max}$ is a maximum. Assume R_{max} is located in slot S_l. If $x(R_{max}) + w_{max} + w_i \leq l \cdot d$, then R_i is assigned position $x(R_{max}) + w_{max}$ in slot S_l. Otherwise it is assigned position $x(B_{l+1})$ in slot B_{l+1}. It is straightforward to see that MIN_CUT(d) generates the number of cuts needed for a given d in $O(n)$ time. Note that when d is less than the width of one of n rectangles, MIN_CUT(d) returns zero.

Assume now that some $l_{i,j}$ is the input for MIN_CUT. If the minimum number of cuts returned is larger than k, a larger width is needed in a configuration making k cut. Otherwise, a minimum configuration making k cuts can possibly achieve a smaller width. Hence, using a binary search, we can find the optimal d^* in $O(\rho + n \log n)$ time.

We next describe an algorithm that solves the minmax problem in $O(n)$ time for $k = 3$. In this case we only have two slots available and a rectangle R_j is either assigned to slot S_1 or slot S_2. The minimum width d^* is now determined by either $l_{0,j}$ or $l_{j,n+1}$ for some j. We thus only determine $l_{0,j}$ and $l_{j,n+1}$ for every j. This is done in $O(n)$ time. An obvious lower bound on the width of a minimum configuration is $\frac{1}{2}l_{0,n+1}$. Let $d_{opt} = \frac{1}{2}l_{0,n+1}$ and let R_j be any rectangle. If $l_{0,j} \leq d_{opt}$, then any left-compressed configuration of minimum width assigns R_j to slot S_1. Otherwise, the decision on where to put R_j is based on the following rule: If $l_{0,j} \leq l_{j,n+1}$, rectangle R_j is assigned to slot S_1; otherwise R_j is assigned to slot S_2. Using these conditions, it is straightforward to develop an $O(n)$ time algorithm. First, we compute $l_{0,j}$ and $l_{j,n+1}$ for each rectangle R_j. We divide the n rectangles into two sets according to the rules stated above and then compute the resulting width d^*. The following theorem is a consequence of the above discussion.

Theorem 4.1 *The minmax k-partition problem can be solved in $O(n)$ time for $k = 3$ and in $O(\rho + n \log n)$ time for $k \geq 4$.*

References

[1] D.G. Boyer. Symbolic layout compaction review. In *Proceedings of 25th ACM/IEEE Design Automation Conference*, pages 383–389, 1988.

[2] Y.E. Cho. Subjective review of compaction. In *Proceedings of 22th ACM/IEEE Design Automation Conference*, pages 396–404, 1985.

[3] K. Melhlhorn. *Data Structures and Algorithm 1: Sorting and Searching.* Springer-Verlag, 1984.

[4] D. A. Mlynski and C. H. Sung. Layout compaction. In T. Ohtsuki, editor, *Layout Design and Verification*, pages 199–235. Elsevier Science Publishers, 1986.

[5] A.R. Newton and A.L. Sangiovanni-Vincentelli. Computer-aided design for VLSI circuits. *Computer*, 19:38–63, 1986.

[6] M. Schlag, F Luccio, P. Maestrini, D. T. Lee, and C. K. Wong. A visibility problem in VLSI layout compaction. In F. P. Preparata, editor, *Advances in Computing Research: VLSI Theory*, pages 259–282, Greenwich, Connecticut, 1984. JAI Press.

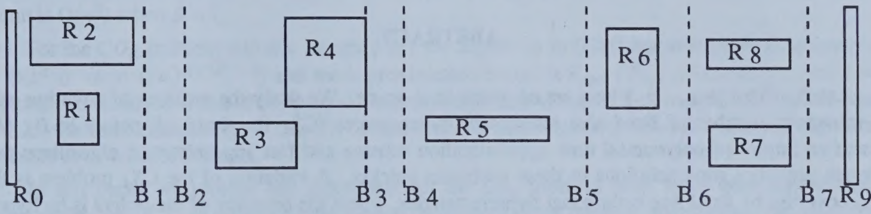

R 2
R 4
R 6
R 8
R 1
R 3
R 5
R 7
R0
B 1
B 2
B 3
B 4
B 5
B 6
B 7
R 9

Fig. 1(a) Initial configuration of a 7-partition problem with 8 rectangles.

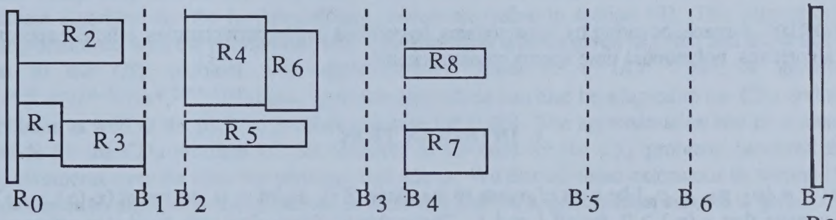

R 2
R 4
R 6
R 8
R 1
R 3
R 5
R 7
R0
B 1
B 2
B 3
B 4
B 5
B 6
B 7
R9

Fig. 1(b) The left-compressed version of the configuration in Fig. 1(a).

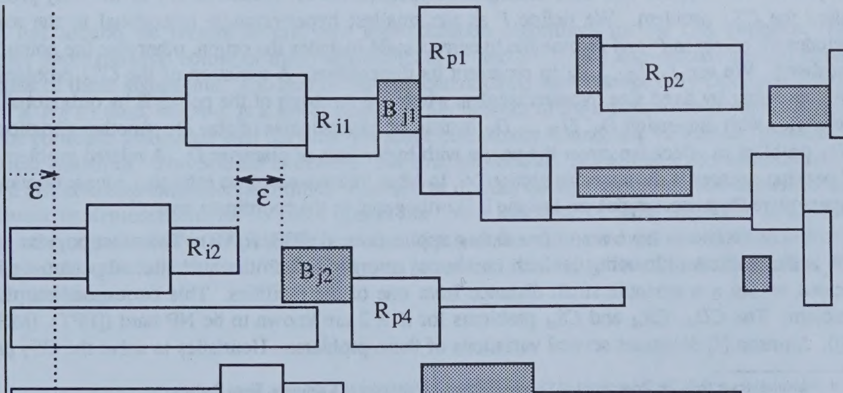

R_{p1}
R_{p2}
R_{i1}
B_{j1}
ε
ε
R_{i2}
B_{j2}
R_{p4}
R_{p3}
R_{p5}

Fig 2. Generate Ca+1 from Ca

COVERING A SET OF POINTS WITH FIXED SIZE HYPERSQUARES AND RELATED PROBLEMS

TEOFILO F. GONZALEZ
Department of Computer Science
Utrecht University
the Netherlands

ABSTRACT

Let $P = \{p_1, p_2, ..., p_n\}$ be a set of points in d-space. We study the problem of covering with the minimum number of fixed size orthogonal hypersquares (CS_d for short) all points in P. We present an improved polynomial time approximation scheme and fast approximation algorithms that generate provably good solutions to these problems quickly. A variation of the CS_d problem is the CR_d, covering by fixed size orthogonal hyperrectangles, where the covering of the points is by hyperrectangles with dimensions $D_1, D_2, ..., D_d$ instead of hypersquares of size D. Another variation is the CD_d problem in which we cover the set of points with hyperdiscs of diameter D. Our algorithms can also be generalized to handle these two problems.

KEYWORDS: d-space, covering by hypersquares, hyperdiscs and hyperrectangles, efficient approximation algorithms, polynomial time approximation scheme.

I. INTRODUCTION

Let $P = \{p_1, p_2, ..., p_n\}$ be a set of points in the plane (E^2). Point p_i is located at $(x_1(p), x_2(p))$, and we assume that $x_j(p_i) \geq 0$, for all i and j. The problem of covering with fixed size orthogonal squares, CS_2, consists of finding a minimum cardinality set of D by D squares covering all points in P, i.e., each point in P must be inside or on the boundary of one of the squares in the cover. A generalization to d dimensions (the set of points belongs to d-dimensional space and instead of covering the points with squares one uses orthogonal hypersquares of dimension D) of the CS_2 problem is called the CS_d problem. We define I as the smallest hyperrectangle orthogonal to the axes that includes all points in P and assume the hyperrectangle includes the origin, otherwise the points can be translated. We use $I_1, I_2, ..., I_d$ to represent its dimensions. A variation of the CS_d problem is the CR_d, covering by fixed size hyperrectangles, where the covering of the points is by orthogonal hyperrectangles with dimension $D_1, D_2, ..., D_d$ instead of hypersquares of size D. Another variation is the CD_d problem in which we cover the points with hyperdiscs of diameter D. A related problem is that of packing squares is discussed in section V. In what follows when we refer to a square (rectangle) or hypersquare (hyperrectangle) we assume it is orthogonal to the coordinate axes.

These problems have many interesting applications ([FPT], [HM]). The most popular application is the problem of locating the least number of emergency facilities such that all potential users are located within a reasonable small distance from one of the facilities. This corresponds to the CD_2 problem. The CD_d, CR_d and CS_d problems for $d \geq 2$ are known to be NP-hard ([FPT], [MIH] and [S]). Johnson [J] discusses several variations of these problems. Heuristics to solve the CS_2 problem

---

‡ On Sabbatical leave from the Department of Computer Science, University of California, Santa Barbara.

have been presented in [T] and [TF]. A polynomial time approximation scheme is given in [HM], i.e., for every constant $\varepsilon > 0$ they give an algorithm that generates solutions such that $F_{apx} / F_{opt} \leq 1 + \varepsilon$ with time complexity $O(n^{O(1/\varepsilon)})$.

For any integer $l \geq 1$, the algorithm for the CS_d problem given in [HM] has time complexity bound $O(l^d \; n^{dl^d+1})$ and the approximation bound is $F_{apx} / F_{opt} \leq (1 + 1/l)^d$. It is important to note that the above bound disagrees with the one in [HM] because there is a typo in that paper. To achieve an approximation bound of 2^d it takes $O(n^{d+1})$ time. To achieve an approximation bound of 2.25, for $d = 2$, it takes $O(n^9)$ time. The only approximation bound that can be guaranteed in practical situations is the bound of 2^d when d is small, since to guarantee a solution within 2^{d-1} the time complexity bound is $O(n^9)$ when $d = 2$.

For the CD_d problem and any integer $l \geq 1$ the algorithm in [HM] has worst case time complexity $O(l^d \; (l \sqrt{d}\,)^d \; (2n)^{d\lceil l\sqrt{d}\rceil^d+1})$ and the approximation bound is $F_{apx} / F_{opt} \leq (1 + 1/l)^d$. For $d = 2$, to guarantee solutions within 4 of optimal, the worst case time complexity bound is $O(n^9)$ and a bound of 2.25 the time complexity is $O(n^{19})$. These large time complexity bounds make the algorithms unusable even when n is small.

In this paper we present fast algorithms for the CS_d problem with worst case approximation bound of 2^d and 2^{d-1}. We also discuss several implementations of these algorithms. The best of the implementations that requires only $O(n)$ space, takes $O(dn + n \log s)$ time, where s is a lower bound on the number of hypersquares in an optimal solution. The constant associated with the time and space complexity bounds are very small, thus usable in practical situations. We also present an efficient algorithm for the l_d-slab problem (which we define in section III). This algorithm can be easily combined with the polynomial time approximation scheme given in [MH] and show that a solution to the CS_d problem with approximation bound $(1 + 1/l)^{d-1}$ can be generated in $O(l^{d-1} \, d(2l^{d-1}\text{-}1)n^{d(2l^{d-1}-1)+1})$ time. Our new algorithms can also be adapted to the CD_d and the CR_d problems as well as the packing problem discussed in [MH]. The approximation and time complexity bounds for the CD_d problem are not identical to the ones for the CS_d problem; however, they are improvements over the ones for previous algorithms. We discuss these extensions in section IV. The approximation algorithms are presented in section II and the approximation scheme is given in section III.

II. APPROXIMATION ALGORITHMS FOR THE CS_d PROBLEM

In this section we present several new approximation algorithms for the CS_d problem. The worst case approximation bound of these algorithms is 2^d and 2^{d-1}. We also discuss several implementations of these algorithms. The best of the implementations that requires only $O(n)$ space takes $O(dn + n \log s)$ time, where s is a lower bound on the number of hypersquares in an optimal solution. The constants associated with the time and space complexity bounds are very small, thus usable in practical situations. This section is divided into four subsection. In each of these subsections we present a different approximation algorithm. Let us now discuss the first approximation algorithm and the different implementations for its basic operations. Note that none of these algorithm is superior over the other algorithms for all problem instances. Each of the algorithms has a domain in which it outperforms the other algorithms. This is why we present all the algorithms together with their different implementations.

(A) MAXIMAL SET OF INDEPENDENT POINTS APPROACH

Before presenting our algorithm based on finding a maximal set of independent points, we define some useful terms. Points p_i and p_j (in P) are said to be *independent* if no hypersquare (with side size D) can cover both of them. Otherwise, we say that point p_i is *dependent* on point p_j or vice-versa. A set $S \subseteq P$ is said to be a *maximal set of independent points* if every pair of distinct points in S is independent and every point in P is dependent on at least one point in S. Whenever we refer to S, a maximal set of independent points, we use s to denote its cardinality.

The idea behind our algorithm is to find a maximal independent set of points. Since no two points in an independent set of points S may be covered by the same square, we know that $F_{opt} \geq |S|$. For the case when $d = 2$, the set of points dependent on each point in set S can be covered by four squares (see figure 1). From this set of squares one can easily delete a maximal set of useless squares, i.e., squares that contain points each of which is inside at least another square. Therefore for $d = 2$, $F_{apx} \leq 4\,|S|$. Combining these two bounds, we know that $F_{apx} \,/\, F_{opt} \leq 4$. For $d > 2$ the approach is similar, but to cover all the points dependent on a point in set S one introduces 2^d hypersquares. Figure 1 shows the cover for the case when $d = 3$. Our first approximation algorithm (MS) to construct the set S is given below.

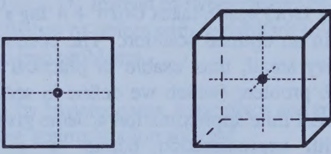


Figure 1: Covering of all points dependent on a point p.

```
Algorithm $MS$ ( $P=\{p_1, p_2, ..., p_n\}, D$ )
  $S \leftarrow \emptyset$;
  for $i = 1$ to $n$ do
    if $p_i$ is independent from every point in $S$ then $S \leftarrow S \cup \{p_i\}$;
  endfor
end of algorithm MS
```

In what follows we discuss several methods to implement the test condition in the **if** statement. Let us consider first a straight forward implementation of the test condition. Testing whether a point is independent from all other points in set S can be accomplished by comparing it against each point in set S. Each test can be accomplished by performing $O(d)$ arithmetic operations and comparisons. Therefore, the overall time complexity for procedure MS is $O(d\ s\ n)$.

The above implementation of the algorithm does not take advantage of properties of the set of points in S. Let us now consider these properties in more detail. We begin with the case when $d = 2$. For any point $p \in S$, all points located at a distance D along each axis from it are dependent on point p. Therefore, the test "if q is independent from each point in S" can be reduced to testing whether or not a point is inside a set of squares with dimensions $2D$ by $2D$, where the center of each of these squares is a point in set S. These squares are called *t–squares (target squares).* The set of *t–squares* is not static, as the algorithm progresses the number of *t–squares* increases. One important property of the *t–squares* is that all of them have identical dimensions ($2D$ by $2D$). Another important property is the fact that the center of a *t–square* is never inside another *t–square*.

Partition the positive quadrant uniformly into squares of size D by D in such a way that a square has its bottom-left corner located on the origin. The squares are called *g–squares (grid squares).* We refer to each *g−square* by a tuple (i, j) for $i, j \geq 0$ (from left to right and bottom-up, with the bottom-left square labeled (0,0)). Since $x_j(p_i) > 0$, the function $i_j(p_i)$ defined as $\lfloor x_j(p_i)/D \rfloor$ can be used to determine the *g−square* to which point p_i belongs. We use $int(p_i)$ to identify the *g−square* to which p_i belongs. Since the *t−squares* have dimensions $2D$ by $2D$ and no center of a *t−square* is inside another *t−square*, there is at most one center of a *t−square* inside a *g−square*. I.e., the maximum number of independent points in each *g−square* is at most one. Determining whether a point p is inside a *t−square* can be tested as follows. First we find the *g−square* $(i_1(p), i_2(p))$ to which it belongs. Then, we construct the set S' of all independent points in S that belong to neighboring *g−squares*, i.e., belong to *g−square* (i'_1, i'_2) such that $|i'_1 - i_1(p)| \leq 1$ and $|i'_2 - i_2(p)| \leq 1$. Clearly, for $d = 2$, there are at most $3^2 = 9$, and in general there are at most 3^d of such neighbors. Since the dimension along each axis of each of the neighboring *g−squares* is D, it then follows that p is independent from each *t−square* iff p is independent from each *t−square* whose center is in set S'.

Let us now define the abstract data type *RD*. The ADT consists of the set T of d-dimensional integer points. The operations to be performed on T are:

INSERT(*int*(p))	[add a point to the set of points T],
DELETE(*int*(p))	[delete a point from the set of points T],
MEMBER(*int*(p))	[test whether or not *int*(p) is in the set T], and
RANGE-1(*int*(p))	[list all points in T that differ from *int*(p) along each axis by at most one unit].

We define algorithm *NMS* as a modified version of algorithm *MS* that uses the *RD* ADT operations. Gonzalez [G] discusses several implementations for the d-dimensional dictionary ADT (*RD* without RANGE-1 queries). Each d-dimensional dictionary operation takes $O(d + \log m)$ time, where m is the number of elements in the set. Any implementation for such ADT can be used for *RD* by simulating each RANGE-1 operation with either 3^d MEMBER operations or s element comparisons. Using Gonzalez' [G] implementation (or an equivalent implementation [G]) for d-dimensional dictionaries results in an algorithm with time complexity $O(\min\{3^d, s\}\ (d + \log s)\ n)$. We call this implementation the *balanced tree* implementation of algorithm *NMS*.

When there is a large amount of memory we may use the double-array representation suggested by Aho, Hopcroft and Ullman [AHU] (exercise 2.12, page 71), in which a set of t integers in the range $[1, w]$ can be represented by two arrays; one of size t and the other of size w; and determining whether an element is present in the set requires constant time. In this case the time complexity bound can be reduced as follows. The set S is represented by a double-array with the d-dimensional elements compressed into a single integer by row-major (or column-major) order. We assume that the maximum value along each dimension is known, otherwise it can be easily computed. A simple formula computable in $O(d)$ time can perform the mapping. The total time reduces to $O(\min\{3^d, s\}\ d\ n)$. It is important to note that the amount of space required in this implementation may be very large ($O(\prod \lceil I_j/D \rceil)$). We call this implementation the *double-array* implementation of algorithm NMS.

One may reduce the additional space at the expense of increasing the time. The idea is to partition the problem by finding connected components and pseudo connected components. For brevity we shall not discuss this further. In a subsequent paper we explain this in detail.

Table I. Time and Space Complexity implementations of algorithm NMS.

implementation	time complexity	space complexity
straight forward	$O(d\ s\ n)$	$O(n)$
balanced trees	$O(\min\{3^d, s\}\ (d + \log s)\ n)$	$O(n)$
double-array	$O(d\ \min\{3^d, s\}\ n)$	$O(\prod \lceil I_j/D \rceil)$

Theorem 1: Algorithm *NMS* generates a solution for the CS_d problem with approximation bound 2^d. The time complexity for different implementations of the algorithm is given in Table I.
Proof: By the above discussion. □

(B) SIMPLE AGGREGATION

This algorithm is the simplest of the four and it generates its solutions quickly. Its worst case approximation bound is identical to the one of the algorithm in the previous subsection and it has a smaller worst case time complexity bound. The only drawback is that the solutions it generates are "normally" inferior, with respect to the objective function criteria, to the one generated by the algorithms in the previous sections. This was our conclusion after 20000 experiments. However, in many instances its solutions are superior.

The algorithm, which we refer to as *SA*, is very simple. First it finds the *g–square* where each point belongs. Then it introduces a square for each *g–square* with at least one point in it. Let $F_{apx} = t$ be the total number of squares introduced. We claim that $F_{opt} \geq t/2^d$ because no square in an optimal solution may have points from more than 2^d of the *g–squares*. Therefore, the approximation bound is 2^d.

Let us now consider different implementations. Finding the *g–squares* where the points belong takes $O(d\ n)$ time. Once we have the *g–squares* where points are located, they can be sorted in $O(dn + n\ log\ t)$ time by the procedures discussed in [G]. Therefore the overall time complexity of this implementation, which we call balanced tree implementation is $O(dn + n\ log\ t)$ time. The other implementation for the second algorithm is similar to the one for the first algorithm. For brevity we shall not discuss them in detail. The performance of our algorithms is summarized in Table II.

Table II. Time and Space Complexity for several implementations of algorithm SA.

implementation	time complexity	space complexity
balanced tree	$O(d\ n + n\ log\ t)$	$O(n)$
double-array	$O(d\ n)$	$O(\prod \lceil I_j/D \rceil)$

Theorem 2: Algorithm *SA* generates a solution for the CS_d problem with approximation bound 2^d. The time complexity for different implementations of the algorithm is given in Table II.
Proof: By the above discussion. □

For any of the implementations discussed for this algorithm there are problem instances that achieve the worst case approximation bound. This algorithm outperforms all the algorithms when t is large and there are many points per *g–square*.

(C) ORDERED MAXIMAL SET OF INDEPENDENT POINTS APPROACH

Let us consider the third algorithm. The idea behind this algorithm is similar to the first one, except that we find a maximal independent set in an orderly fashion. Let us consider first the case when $d = 2$. We sort the points with respect to their x-coordinate values (in case of ties the order is not important). We shall traverse the points in that order. Our third approximation algorithm (*OMS*) to construct the set S is given below.

```
Algorithm $OMS(P, D)$
  Rearrange the set of points with respect to their first coordinate value;
  // I.e., $x_1(p_1) \le x_1(p_2) \le ... \le x_1(p_n)$ //
  $S \leftarrow \emptyset$;
  for $i = 1$ to $n$ do
    if $p_i$ is independent from every point $p$ in $S$ then $S \leftarrow S \cup \{p_i\}$;
  endfor
end of algorithm $OMS$
```

As in the case of the analysis of procedure MS, it is simple to show that $F_{opt} \ge |S|$. For the case when $d = 2$, the set of points dependent on a point p in S that are not located to its left can be covered by two squares as shown in figure 2. It is important to note that when considering point p all the points previously visited can be covered by adding two squares for each point in the current maximal independent set. Therefore, $F_{apx} \le 2\,|S|$. Combining these two bounds, we know that $F_{apx} / F_{opt} \le 2$. For $d > 2$, the approach is similar, but to cover all the points dependent on a point in set P one introduces no more than 2^{d-1} hypersquares. Figure 2 shows the cover for the case when $d = 3$.

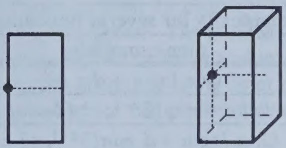


Figure 2: Covering the set of points dependent on p_i and located to the right of p_i.

The straight forward implementation of *OMS*, which is similar to the one for *MS* has overall time complexity $O(n \; log \; n + d \; s \; n)$. The above implementation of the algorithm does not take advantage of special properties of the sets we are dealing with. Let us now consider these properties in more detail.

As with the case of algorithm *MS*, there are several implementations for algorithm *OMS*. The implementations are similar to the ones in the previous case. The main difference is that instead of *t–squares* we have *t–rectangles* with dimensions D by $2D$ (rather than $2D$ by $2D$). Also, when considering point p we only need to consider those *t–rectangles* whose distance (along the first coordinate value) is at most D from it. Therefore, the cardinality of the set of *t–rectangles* may increase or decrease as the algorithm advances. Remember that for algorithm *MS* the cardinality of the set of *t–squares* only increased in size, there were no deletions. The "center" of a *t–rectangle* is the point on the left boundary that is equidistant to the left corner points of the rectangle. We define the *g–squares* as a strip of width D that is constantly moving from left to right, in such a way that the current point is located on its boundary. Only those *t–rectangles* whose "center" is in this strip need to be considered. The *g–squares* are denoted by integer (i), for $i \ge 0$ (the bottommost *g–square* is labeled 0). Since all of the *t–rectangles* have dimensions D by $2D$ and no "center" of a *t–rectangle* is inside another *t–rectangle*, there is at most one "center" of a *t–rectangle* inside a *g–square*. I.e., the maximum number of independent points in each *g–square* is at most one. Determining whether a point p is inside a *t–rectangle* can be determined as follows: first we find the *g–square* (i) to which p belongs. Then, we construct the set S' of all independent points that belong to neighboring *g–squares*, i.e., belong to *g–square* (i') such that $|i'-i| \le 1$. For $d = 2$, there are 3, and in general there are 3^{d-1} of such neighbors. Since the x-length and the y-length of each of the neighboring *g–squares* is D, it then follows that p is independent from each *t–rectangle* iff it is independent from each *t–rectangle* whose "center" is in set S'.

Let us now explain how to implement the above test efficiently. We define the set of elements S, such that *g–square* (i') has a "center" of a *t–rectangle* inside it iff $i' \in S$. Associated with each element $k \in S$ there a point, denoted by t_k that represents the "center" of the *t–rectangle* inside *g–square* (k). Given a point $p_i \in P$, by simple arithmetic operations we determine (i'), the *g–square* that contains p_i. Then we search in the set S for the elements that differ from i' by at most 1. Once we have determined which *g–squares* in the neighborhood have a "center" of a *t–rectangle* inside them, we test whether or not p_i is inside any of those *t–rectangles*. If the answer is yes, we just proceed to the next point, otherwise, p_i is independent from all points in S, so we have found a new *t–rectangle*. Remember that its "center" belongs to *g–square* (i'). Therefore, we add (i') to S. We also associate p_i with the entry i' and proceed to point p_{i+1}. At this point we delete the *t–rectangles* that do not belong to the new grid. Deletion in done in the FIFO fashion. The worst case time complexity for the new method is similar to the one in (A), though it is not hard to see that in a large number of problem instances the third algorithm would be faster.

The other implementation for the third algorithm is similar to the one for the first algorithm. For brevity we shall not discuss them in detail. The performance of our algorithms is summarized in Table III.

Table III. Time and Space Complexity for several implementations of algorithm OMS.

implementation	time complexity	space complexity
straight forward	$O(n \; log \, n + d \; s \; n)$	$O(n)$
balanced tree	$O(n \; log \, n + \min\{3^{d-1}, s\} \, (d + log \, s) \, n)$	$O(n)$
double-array	$O(n \; log \, n + d \; min\{3^{d-1}, s\} \, n)$	$O(\prod \lceil I_j / D \rceil)$

Theorem 3: Algorithm *OMS* generates a solution for the CS_d problem with approximation bound 2^{d-1}. The time complexity for different implementations of the algorithm is given in Table III.
Proof: By the above discussion. □

(D) PARTITION INTO INDEPENDENT SUBPROBLEMS APPROACH

The fourth algorithm is slightly different than the previous ones. Its approximation bound is identical to the one for the third algorithm, but it can be implemented to run faster. Before we explain it, let us consider a restricted version of it which we call the *slab problem*, i.e., all points are located inside a rectangle (whose sides are orthogonal to the axes) with height D. An optimal solution to the slab problem can be obtained by projecting all points to the bottom side of the rectangle and applying an optimal algorithm for the one dimensional case. An optimal algorithm for the one-dimensional problem considers all points in increasing order. The leftmost point is covered by the leftmost end point of a line segment of length D. All the points that fall inside this line segment are covered by the same line. Then the leftmost uncovered point is covered in a similar fashion. This procedure, referred to as the *cover the leftmost point first (CLPF)*, has $O(n \; log \, n)$ time complexity.

The CLPF procedure is not the procedure with the best worst case time complexity bound to find an optimal solution to the slab problem. Let us now briefly discuss a better algorithm (**fastCLPF**) whose worst case time complexity bound is only $O(n \; log \, s)$, where s is the number of elements in an optimal cover. The idea behind the procedure is to sort the elements with respect to their $i_1(p)$ values. The elements are placed in sets $S_1, S_2, \ldots, S_k$, where all elements in set S_i have identical $i_1(p)$ values and the $i_1(p)$ value of the elements in set S_i is smaller than those in S_{i+1}. Clearly, no element in S_i can be with an element in S_{i+2} in the same square in a feasible solution. The idea behind the algorithm is to introduce a square for the leftmost point in S_1. Then the points in S_1 and S_2 that are covered by that square are deleted. Then we select the leftmost point in S_2 (if any) and so on.

Now let us use the above procedure to obtain a fast approximation algorithm for the CS_2 and then for the CS_d problem. When $d = 2$, the set of points is partitioned into two sets, R_1 and R_2. For each of these two problems we find an optimal cover. For simplicity, let us explain our algorithm in terms of the *g-squares* introduced for the first algorithm. All the points that belong to the *g-squares* (i,j) such that j is odd belong to problem R_1 and the ones such that j is even belong to subproblem R_2. Let us consider problem R_1. It is simple to see that all the points in R_1 belong to slabs (with height D) and which are D units apart. Therefore, an optimal solution to R_1 consists of finding an optimal solution to each of the slabs which we know can be done in $O(n \log s)$.

Procedure PARTION-FIRST partitions the elements with respect to $(i_2(p), ..., i_d(p))$ into sets and then applies procedure fastCLPF to each set. The partition can be done by sorting the elements. Sorting can be done by the algorithm in [G] (or an equivalent algorithm) in $O(d\ n + n \log s)$ time. The second part applies procedure fastCLPF which takes overall time $O(n \log s)$. Therefore the overall time complexity of our procedure is $O(d\ n + n \log s)$. We call this implementation the *balanced tree* implementation. Let us now consider the *double array* implementation of procedure PARTITION_FIRST. By using the double-array structure one can find the partition of the elements in $O(d\ n)$ time. Procedure first-CLPF can also be implemented in $O(d\ n)$ time by using double arrays. The idea is to first use the double-arrays to partition the elements into classes that have the same $i_1(p)$ value. Then "adjacent" classes are grouped together. The problem defined over each group of classes can be solved independently by applying the fast-CLPF procedure. The outcome would be the same as in the previous implementation, the only difference is the order in which the output is generated by the algorithm. Table IV lists the time and space complexity for several implementations of our procedure.

Table IV. Time and Space Complexity for algorithm PARTITION_FIRST.

implementation	time complexity	space complexity
balanced tree	$O(d\ n + n \log s)$	$O(n)$
double-array	$O(d\ n)$	$O(\lceil I_1/D \rceil + \prod_{j=2}^{d} \lceil I_j/D \rceil)$

Theorem 4: Algorithm PARTITION-FIRST generates a solution for the CS_d problem with approximation bound 2^{d-1}. The time complexity for different implementations of the algorithm is given in Table IV.

Proof: By the above discussion. □

III. POLYNOMIAL TIME APPROXIMATION SCHEME

Let us now consider the *l_2-slab* problem, i.e., all points lie in E^2 inside a rectangle with height lD, for some integer $l \geq 1$. First we show that the l_2-slab problem can be solved in $O(4l\, n^{4l})$ time via dynamic programming. The dynamic programming procedure finds all covers that satisfy certain special properties (which we specify later on). We then generalize the method to solve the l_d-slab problem (i.e., all points lie in E^d inside a hyperrectangle in which all dimensions, except the first, have length lD). Finally, we present an improved polynomial time approximation scheme by showing how to combine the algorithm for the l_d-slab problem with the polynomial time approximation scheme in [HM].

Let us assume that the set of points are sorted with respect to their first coordinate value, i.e., $x_1(p_1) \leq x_1(p_2) \leq \dots \leq x_1(p_n)$. A square in c_i in a cover C is said to be an *a–square (anchored square)* if there is a point r in P located on the left side of c_i and a point s in P (not necessarily different than r) located on the top side of c_i that are not contained (are not located inside or on the boundary) in any other square in the cover C. For an *a–square* c_i in cover C, the point p_j with least index located on the left (top) boundary of c_i which is not contained in any other square in C is called the *left (top) anchor of c_i in C*. We shall also refer to the two points as simply *anchors*. A cover is said to be an *a–cover (anchor cover)* if all the squares in it are *a–squares*. It is simple to show that any cover can be transformed to an *a–cover* without increasing the number of squares in it. We say that an *a–cover* is a *s–cover* if every vertical line does not partition into two nonempty parts more than $2l$-1 squares in the cover. To show that the time complexity of our algorithm is bounded by a polynomial on n, we prove the following lemma which states that every problem instance has an *s–cover* among the set of optimal covers.

Lemma 1: Every l_2-slab problem has an *s–cover* among the set of optimal covers.
Proof: The proof is based on interchange arguments. ☐

Before presenting our algorithm we need to make a few more definitions. Deleting a subset of squares from an *a–cover* C results in a *partial a–cover* which we call C'. If points $p_1, p_2, \dots p_i$ are covered by a *partial a–cover* C', and all the left anchors of the squares in C' are points from the set $\{p_1, p_2, \dots, p_i\}$, then C' is called an *i-partial a–cover*. Two *i-partial a–covers* are said to be *equivalent* if the left and top anchors of every square partitioned into two nonempty parts by any vertical line "immediately to the right" (any line between p_i and p_j, where j is the smallest index of a point that appears to the right of p_i) of p_i are identical in the two *i-partial a–covers*. The *i-partial a–cover C dominates i-partial a–cover C'* iff C and C' are equivalent and C has fewer squares that C'. An *i-partial a–cover* is said to be an *optimal i-partial a–cover* if there is an *s–cover* in the set of optimal covers with an *i-partial a–cover* equivalent to it. Procedure **DP** processes the points in the order $p_1, p_2, \dots, p_n$. During the ith iteration we construct the set of all irreducible *i-partial a–covers*, i.e., set of all *i-partial a–covers* such that no one of them dominates another and no two of them with the same number of squares are equivalent. To show correctness we prove that all optimal *i-partial a–covers* are in the set of irreducible *i-partial a–covers*.

Lemma 2: For every l_2-slab problem, algorithm DP generates an optimal cover in O($4l\ n^{4l}$) time.
Proof: For brevity the proof is not included. ☐

For the l_d-slab problem, the number of square partitioned by a hyperplane passing through p_i is at most $2l^{d-1}$-1 instead of $2l$-1, and the number of anchors per square is d instead of two. Therefore, the time complexity for the l_d-slab problem is O($d(2l^{d-1}-1)n^{d(2l^{d-1}-1)+1}$).

This algorithm can be easily incorporated with the polynomial time approximation scheme given in [HM]. The idea is to apply the algorithm to l^d problem instances of the l_d-slab problem. The l^d problem instances can be partitioned into l^{d-1} groups of problems such that in each group the number of points in all the problems is n. This results in an overall time complexity bound O($l^{d-1}\ d(2l^{d-1}-1)n^{d(2l^{d-1}-1)+1}$). The approximation bound is $(1+1/l)^{d-1}$.

Theorem 5: Combining algorithm *DP* with the polynomial time approximation scheme in [HM] results in a procedure that generates a solution to the CS_d problem with approximation bound $(1+1/l)^{d-1}$ and time complexity O($l^{d-1}\ d(2l^{d-1}-1)n^{d(2l^{d-1}-1)+1}$).
Proof: By the above discussion. ☐

IV. DISCUSSION

All our techniques can also be adapted to the CR_d and the CD_d problem. For the CR_d problem the approximation and time complexity bounds are identical. For the CD_d problem the algorithms in section II have identical time complexity bounds; however, the approximation bounds are $\lceil 2\sqrt{d}\rceil^{d}$, $\lceil 2\sqrt{d}\rceil^{d}$, $\lceil 2\sqrt{d}\rceil^{d-1}\lceil \sqrt{d}\rceil$, and $2^{d-1}\lceil \sqrt{d}\rceil^{d}$, respectively. For the polynomial time approximation scheme, the approximation bound is $2(1 + 1/l)^{d-1}$ and the time complexity bound is $O(l^{d-1} d \lceil 2\sqrt{d}\rceil \lceil l\sqrt{d}\rceil^{d-1} n^{d\lceil 2\sqrt{d}\rceil^{d-1}+1})$. The same techniques can be adapted to the packing problem studied in [MH]. The approximation and time complexity bounds are smaller than for the CS_d problem. For brevity we do not discuss this further. As pointed out in [HM] it is important to develop fast approximation algorithms for the l_d-slab problem. These algorithms would imply a much better time complexity bound for the CS_d problem at the expense of a slightly larger approximation bound.

From our algorithms one can define a heuristic which we believe has a reasonable behavior with respect to the time complexity and the approximation. The idea is to assign the points to the *g*–*squares* first. Then the *g*–*squares* with points are partitioned into two groups. Those with a single point and those with at least one point. The heuristic then applies our polynomial time approximation scheme to the problem defined by all the points belonging to single point *g*–*squares*. An optimal solution will be generated quickly for this subproblem. The remaining problem is solved by any of the approximation algorithms in section II.

V. REFERENCES

[AHU] Aho, A., J. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1975.

[FPT] Flower, R. J., M. S. Paterson, and S. L. Tanimoto, "Optimal Packing and Covering in the Plane are NP-complete," Information Processing Letters 12, 1981, 133 - 137.

[G] Gonzalez, T., "The On-Line *d*-Dimensional Dictionary Problem", Technical Report, University of Utrecht, July, 1990.

[HM] Hochbaum D. S. and W. Maass, "Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI," *Journal of the ACM*, Vol. 32, No. 1, January 1985, 130 - 136.

[J] Johnson, D., "The NP-Completeness Column: An Ongoing Guide," *Journal of Algorithms* 3, 182 - 195, 1982.

[MIH] Masuyama, S., T. Ibaraki, and T. Hasegawa, "The Computational Complexity of the m-center problems on the Plane," *Transactions IECE of Japan* E64, 1981, 57-64.

[S] Supowit, K. J., "Topics in Computational Geometry," Report No. UIUCDCS-R-81-1062, Department of Computer Science, University of Illinois, Urbana, Ill., 1981.

[T] Tanimoto, S. L, "Covering and Indexing an Image Subset," Proceedings of the 1979 IEEE Computer Society Conference on Pattern Recognition and Image Processing, 239 - 245, 1979.

[TF] Tanimoto, S. L. and R. J. Fowler, "Covering Image Subsets with Patches," Proceedings of the 5th International Conference on Pattern Recognition, 835 - 839, 1980.

A Simple Strategy for Deadlock Prevention in Store-and-forward Networks

S.W. Lau & F.C.M. Lau
Department of Computer Science
University of Hong Kong

October 1990

Abstract

We propose and discuss a simple strategy for deadlock-prevention in store-and-forward networks. The strategy is based on the idea of allocating empty buffers to cycles of nodes. Since the number of cycles could be large, we propose a method for reducing the number of cycles that require an empty buffer, which could lead to a substantial saving in storage space. We compare our strategy with other deadlock schemes and show that ours is better with respect to some desired properties.

1 Introduction

We are interested in deadlock prevention in arbitrary (store-and-forward) networks. In these networks, the existence of deadlocks is due to the cyclic waiting of packets for buffers. Our method prevents deadlocks by insuring that cyclic waiting would never occur. This is achieved by maintaining a "propagating" empty buffer in every cycle of nodes (hence a chain of waiting packets is always short of at least one packet to form a cycle). Unfortunately, unless the network is very sparse, the number of cycles, and hence the number of empty buffers required, could be very large. In this work, we propose a simple way of reducing this buffer space requirement by supplying empty buffers to a selected subset of cycles, with the unselected cycles being some kind of composite of these selected cycles. With the network nodes executing three simple, special algorithms, the network is proved to be deadlock-free with this reduced number of empty buffers.

2 Basic Ideas

We represent a store-and-forward network by a connected graph $G = (V, E)$ where an edge $e \in E$ represents a two-way communication link and a vertex $v \in V$ represents a node of the network. We consider networks with the following properties: (1) each node of the network has a limited number of buffers, (2) each packet occupies one buffer in a node, (3) a packet which has arrived at its destination node will be consumed in finite time, and (4) the network is static, and all nodes and links are without faults [1].

---

[1] Our algorithm can be easily extended for dynamic networks.

Because of property (1), store-and-forward deadlocks may occur when there is a closed chain of packets waiting for buffers in a cycle[2] of nodes—more exactly, a cycle of filled buffers. A deadlock is direct if the cycle involves only two packets; otherwise it is indirect. Suppose we always maintain at least one empty buffer in a cycle of nodes, then deadlocks would not occur as a cycle of filled buffers cannot exist (there is always a hole in the chain!). If we do this to every cycle in the network, the network is deadlock-free.

The deadlock prevention scheme we propose in this paper is based on the above idea of supplying empty buffers to cycles of nodes. Four questions are in order: (1) how to maintain the empty buffer in a cycle? (2) what is the space complexity, due to empty buffers, of this scheme, and how can it be reduced? (3) what is the time complexity of this scheme? (4) what is the performance of this scheme? In this paper, we focus on (2) and suggest a strategy to cut down on the number of cycles that require an empty buffer. We comment on (3) briefly at the end of this paper. (4) will be the subject of another paper. We first discuss the basic protocol for maintaining an empty buffer in a cycle.

We omit the detail of how an empty buffer is generated in the first place and concentrate on how it is maintained at runtime. The protocol is simple: *let the empty buffer propagate.* Precisely, when a node v sends a packet to an adjacent node in the cycle, say w, and the packet falls into the empty buffer which happens to be in w, then the buffer vacated by v is marked to be the empty buffer (the empty buffer has propagated from w to v). Note however that a node currently holding the empty buffer would not permit a local process to put a message in the empty buffer since this would consume the empty buffer. Hence, if a node is passed through by exactly one cycle, the minimum number of buffers the node needs to have is two: one for admitting messages from local processes (a message in this buffer will eventually be forwarded, given that the network is deadlock-free), and the other one for the propagation of the empty buffer. Note moreover that an empty buffer cannot be shared by more than one cycle, and hence the lower bound on the total number of buffers in the network is given by

$$\sum_{v \in V} \mid \text{cycles passing through } v \mid +1$$
$$= \text{the total length of cycles} + \mid V \mid.$$

The naive looking example in Figure 3 has 20 cycles, and the total length is quite considerable when compared with the number of nodes or edges.

Our discussion so far applies to indirect deadlocks involving more than two nodes and more than one link. However, since a link is bidirectional, it is a degenerate case of a cycle, and if we were to supply an empty buffer for every link, the network would suffer from a profusion of empty buffers. Instead, we devise a simple algorithm to prevent direct deadlocks that occur over a single link. The next section presents this algorithm as well as an algorithm for the propagation of empty buffers.

3 Basic Algorithms

The first algorithm[3] is executed by every node in the network for the prevention of direct deadlocks.

---

[2] One with no vertices being visited more than once.

[3] A similar algorithm can be found in [8].

Algorithm AX *"Atomic Exchange". When a node v_1 receives a request of admission of a packet m_2 from an adjacent node v_2, has a packet m_1 needed to be sent to v_2, and has an ID $\prec$ the ID of v_2, it sends an "exchange" request to v_2. Upon receipt of the exchange request, v_2 sends m_2 to v_1; v_1 stores m_2 in an auxiliary buffer and sends m_1 to v_2; v_2 puts m_1 into the buffer vacated by m_2; after sending m_1, v_1 copies m_2 into the buffer vacated by m_1.*

The algorithm is feasible if we assume the network is equipped with a lower-level protocol for the exchange of control packets like the admission and exchange requests in the above algorithm. With the ID's and the "$\prec$" relationship, only one of the two adjacent nodes would initiate the exchange. Since the network is static, generating and assigning these ID's is a trivial matter. By selecting a subset of nodes that can initiate an exchange, the total number of auxiliary buffers needed can be minimized. Using Algorithm AX, the lower bound on the total number of buffers becomes

$$\text{the total length of cycles} + \mid V \mid + min(\mid V \mid -1, \mid E \mid /2)$$

The next algorithm is for maintaining the empty buffer in a cycle. It realizes the protocol for empty buffer propagation we described in the previous section.

Algorithm BP *"Empty Buffer Propagation". Given any cycle of nodes c. If a node $v_1 \in c^4$ holding the empty buffer receives a request of admission of a packet m_2 from an adjacent node $v_2 \in c$ and decides to use the empty buffer for m_2, it notifies v_2 with a control packet. Upon receipt of this control packet, v_2 sends m_2 to v_1 and marks the buffer so vacated as the empty buffer for c. Upon receipt of m_2, v_1 puts it in the original empty buffer which v_1 now unmarks.*

It is now clear that if every node executes these two algorithms, and provided that the number of buffers is right, deadlocks (direct and indirect) are prevented, as summarized by the following theorem.

Theorem 1. *For an arbitrary network, if (a) every cycle (length > 2) has an empty buffer, (b) each node has at least the number of buffers equal to 1 (2 if the node can initiate an atomic exchange) + the number of cycles passing through the node, and (c) each node uses Algorithms AX and BP, then the network is deadlock-free.*
The proof of this theorem is straightforward.

4 Cycles Reduction

Now, by way of an example, we demonstrate how it is possible to reduce the number of cycles that require an empty buffer. The example in Figure 1 shows a network with three intersecting cycles.[5] Obviously, if the network satisfies the conditions of Theorem 1—i.e., to provide three empty buffers to the three cycles respectively—then the network is free of deadlocks. Now the question is: can we use fewer empty buffers?

---

[4] $v \in c$ means v is a vertex of the cycle c.

[5] Two cycles *intersect* iff they have a common edge. Note also that in the example any one of the cycles is a *ring sum* [13] of the other two cycles.

Consider the situation in which we provide only two empty buffers, b_1 and b_2, to the two smaller cycles, c_1 and c_2, respectively, but none to c_3. Then deadlocks may occur in c_3, in the following cases:

Case 1: the two empty buffers are in v_7, and v_1 and v_2 have no messages to send to v_7;

Case 2: one of the empty buffers is in v_7, and the other empty buffer is in v_1 (or v_2), and v_1 and v_2 have no messages to send to v_7;

Case 3: b_1 is in v_1 and b_2 in v_2 (or vice versa).

In the last case, a deadlock in c_3, if exists, would be in the counter-clockwise direction as v_1 would not accept a message from v_4 (similarly for v_2 and v_6). Whether we can make this network, which is now short of one empty buffer, deadlock-free depends on how we treat these cases.

Consider Cases 1 and 2: if v_7 does not exist, then these two cases would be impossible and hence no deadlocks. Case 3 is still possible, but if we could swap the two empty buffers, then the problem is solved (v_4 can then send to v_1 which now possesses the empty buffer b_2, and so on). Hence, if we invent an algorithm to detect Case 3 and swap the empty buffers, and if all the cycles intersect at at most one edge (we call this a common edge), then we can do away with the empty buffer that would otherwise be needed for any cycle (e.g., c_3) that is a ring sum of these intersecting cycles, and still have a deadlock-free network. The following is such an algorithm. Unlike the previous two algorithms which are executed by every node, this algorithm is executed only by the nodes of a common edge.

> **Algorithm BX** *"Empty Buffer Exchange". Let c_1 and c_2 be two cycles that intersect at one edge, and b_1 and b_2 respectively be their empty buffers. Suppose, referring to Figure 2, b_1 is now in v_1, and v_1 receives a request of admission of a message m from v_4. If v_1 has no available buffers to accommodate m, it sends a control packet to v_2 to request for a possible exchange of empty buffers. Upon receipt of this control packet, v_2, if possesses the other empty buffer, b_2, would exchange it for b_1 with v_1.*

As we are interested mainly in how we would swap the empty buffers when Case 3 above surfaces, we left out the detail of what would happen if v_2 does not possess the other empty buffer.

We are now ready to put the three algorithms together to establish the complete strategy for deadlock-freedom. We do this in the next section and in a more formal way.

5 Putting the Algorithms Together

First, we need to be specific about the choice of cycles that are candidates for omitting the empty buffer.

Definition 1. *The intersection graph of a set of cycles C is an undirected graph in which a node represents a cycle in C and an edge represents a common edge between two (intersecting) cycles in C.*

Definition 2. *A cycle c is the 2-c ring-cover of a set of cycles C if (a) c is formed by deleting all the common edges of the cycles in C, (b) the intersection graph of C is connected and acyclic, and (c) each node in c is passed through by at most two cycles in C.*

If we delete v_7 from Figure 1, then c_3 is an example of a 2-c ring-cover, and $C = \{c_1, c_2\}$. We can easily prove that following properties are true of c_3 as well as any 2-c ring-cover.

Lemma 1. *If c is the 2-c ring-cover of a set of cycles C, then (a) any two cycles in C have at most one common edge, and (b) c passes through all the nodes that are passed through by the cycles in C.*

The next lemma affirms the utility of Algorithm BX.

Lemma 2. *Let c be a 2-c ring-cover of a set of cycles C. If (a) each cycle in C has an empty buffer, (b) each node in c uses Algorithms BP, and (c) each node on a common edge uses Algorithm BX, then indirect deadlocks would not occur in c.*

Sketch of Proof: We assume an indirect deadlock occurs in c. Then we show that only those nodes that are passed through by two cycles may have empty buffers (otherwise, this indirect deadlock would not have occurred). By Definition 2, there are $| C | -1$ intersections, and by (a) of the Lemma, there are $| C |$ empty buffers; hence, there must exist an intersection (between two cycles) that has two empty buffers. By Lemma 1, this intersection has one common edge, and hence the only possible scenario in which this deadlock in c may exist is the one reflected in Case 3 in the example we discussed before. But since the two nodes in c would execute Algorithm BX when this happens, this deadlock is impossible. QED

Putting the three algorithms together, we have this final theorem concerning the deadlock-free property of the network:

Theorem 2. *Given a network G and a (selected) subset of cycles C in G. If (a) each cycle in C has an empty buffer, (b) each cycle in G is either in C or a 2-c ring-cover of a subset of C, (c) each node in G has at least the number of buffers equal to 1 (2 if the node can initiate an atomic exchange) + the number of cycles in C passing through the node, (d) each node in G uses Algorithms AX and BP, and (e) each node on a common edge uses Algorithm BX, then the network is deadlock-free.*

Proof: By Lemma 2 and conditions (a)–(e). QED

Given that our main concern is to minimize the storage space requirement (due to empty buffers), it is important that we select a subset of cycles C such that their total length is sufficiently small. We outline a feasible selection procedure: starting from a set of fundamental cycles C_1 [9] and the set of the remaining cycles C_2 in G, we select a circuits $c \in C_2$ and check whether c is a 2-c ring-cover of a subset of cycles of C_1 (this can be done in time proportional to the number of nodes in c). If c is not a 2-c ring-cover, c is added into C_1. Then c is deleted from C_2. This is repeated until $C_2 = \phi$, and the final C_1 is the required subset of cycles C. This simple procedure, however, may not yield an optimal total length of cycles. If the total number of buffers required is needed to be optimized, we need to minimize the total length of the cycles in C. For the case that C is a set of fundamental cycles, the problem is NP-complete [10]. Nonetheless, an efficient algorithm of $\Theta(| V | + | V | \times | S(G) |)$ time complexity for finding $S(G)$ for planar graphs has been proposed in [12], where $S(G)$ is the set of cycles in G. If the graph is not planar, the algorithm described in [11] can be used instead.

Figure 3 shows an example of a simple network. With our algorithms, the number of cycles that require an empty buffer can be reduced from 20 to 8.

6 Livelock Prevention

Livelock is caused by unfair sharing of buffers among the packets in the network. For example, if packets from a node sent through a communication link in the LIFO (last-in-first-out) order, then the packet that arrived at this node the earliest may be blocked forever (livelocked) if other packets keep coming in without ceasing. The necessary conditions for the occurrence of a livelock are as follows.

Lemma 3 *If a packet m is livelocked in a node v, then at least one of the following conditions is true:*

1. *$\exists$ time t such that after time t, m is sent to an adjacent node v' only when there is no other packet in v needed to be sent to v'.*
2. *$\exists$ time t such that after time t, the next node of the route of m, say v', accepts packets from v only when there are no other packets in the adjacent nodes of v' to be sent to v'.*

By eliminating the necessary conditions for livelock, the following algorithm prevents any occurrence of livelock.

> **Algorithm LP** *"Livelock Prevention" $\forall v \in V$, if there are a set of packets in v needed to be sent to an adjacent node $v' \in V$, v sends the packets one by one in the order of their arrival at v'. $\forall v \in V$, and for each node v' adjacent to v, v records the last time v stored a packet from v' in a buffer. Let this time be denoted by $t(v, v')$.*[6] *When v has a vacant buffer to accept a packet from a set of nodes V', v accepts a packet from the node in V' for which $t(v, v')$ is smallest. When a node $v \in V$ runs out of vacant buffers, it either uses Algorithm AX to exchange a packet with an adjacent node or uses a (marked) empty buffer to accept a new packet.*

The above algorithm guarantees that the buffers in the network are shared equally among the packets by enforcing a first-in-first-out order.

Theorem 3 *If (a) the conditions of Theorem 1 or 2 are satisfied and (b) each node $v \in V$ uses Algorithm LP, then the network is deadlock- and livelock-free.*

Proof: We only present the proof of this Theorem for the conditions of Theorem 1 as the proof for the conditions of Theorem 2 is similar. By (i) and Theorem 1, G is deadlock-free. Suppose there exists a packet m_0 in $v_0 \in V$ which is livelocked. By Lemma 3, condition 1 or condition 2 is true. By (b), packets are sent in the order of their arrival; therefore, only condition 2 of Lemma 3 is true—i.e., there exists time t_0 such that v_0 cannot send any packet to v_1 where v_1 is a node blocking m_0. By (b) of Theorem 1, there always exists a buffer that is not marked to be empty in the buffer pool of v_1. Therefore, after some time $t'_0 \geq t_0$, all buffers, except empty buffers, in v_1 are permanently filled with packets; otherwise, by (b), v_1 will accept a packet from v_0 using a vacant buffer. We have two possible cases:

---

[6] Assume that v records the time $t(v, v')$ by using its monotonically increasing clock which clicks at every acceptance of a packet.

Case 1: There exists a packet m_1 in v_1 such that m_1 is livelocked.
Case 2: All packets in v_1 are not livelocked.

Suppose Case 2 is true. Since all packets in buffers of v_1 are not permanently blocked, then after time t'_0, packets in v_1 are sent to the adjacent nodes of v_1 or consumed in v_1. Therefore, by (b), v_1 will eventually have a vacant buffer to accept a packet from v_0. This leads to a contradiction, and hence this case cannot be true.

Now, suppose Case 1 is true. Let v_2 be one of the nodes that block m_1. By the same argument above, there exists a time $t'_1 \geq t'_0$ such that after t'_1, all buffers, except empty buffers, in v_2 are permanently filled with packets.

Continue in this manner, we have a sequence of nodes $v_1, v_2, \ldots$ in which, after some time t, all the buffers in v_i $(i = 1, 2, \ldots)$, except empty buffers, are permanently filled with packets. The nodes are all distinct,; otherwise, a deadlock occurs in a set of nodes in G. The sequence must be finite since the number of nodes in G is finite. Let v_k be the last node in the sequence. v_k has a packet m_k in one of its buffers. Since v_k is the last node, m_k has arrived at its destination. Therefore, m_k will be consumed after a finite period of time, and a buffer will be released in v_k. This leads to a contradiction. Therefore, livelocks cannot occur. QED

7 Related Works and Comparison

From the point of view of practicality and performance, an ideal algorithm for store-and-forward deadlock prevention should have the following properties: minimal partitioning of the buffer pool in a node, minimal restrictions on routing, livelock freedom, no discarding (and retransmission) of packets, efficient handling of packets at every node, and tunable for meeting storage constraints. Existing algorithms do fall short of having some or all of the these properties. With the algorithms proposed in [1-5], the number of required buffers in each node grows linearly with the diameter of the network (or the maximum distance of the packet routes for cyclic adaptive routing). The FS and BS algorithms described in [2] and [3] require $\Theta(d^2)$ operations (additions and comparisons) to handle a packet, where d is the distance traveled by the packet, and hence not efficient. Likewise, the FSET algorithm in [3] requires at least $\Theta(d^2)$ comparisons and $k \times d$ comparisons in the worst case to handle a packet, where k is the maximum length of a packet route. The algorithm proposed in [6] has to restrict the number of possible routes that a packet can travel in order to prevent deadlocks. The algorithm described in [7] requires retransmission of packets that are discarded because of a buffer being needed by a higher priority packet, and this prioritisation is imposed by the deadlock prevention scheme. The prevention algorithm proposed in [8] is limited to acyclic networks. We claim that our strategy is a better one with respect to the desired properties stated above. Using our strategy, the total number of required buffers B in the network is $L(C) + O(\mid V \mid)$, where C is a selected subset of cycles and $L(C)$ is the total length of the cycles in C. If G is sparse, the number of cycles is small and our algorithm requires fewer buffers than the FS, BS and FSET algorithms [2,3]. With our algorithms, the only restriction as far as buffer usage is concerned is that empty buffers cannot be used indiscriminately to store any packets. The ratio of the number of empty buffers to the total number of buffers is $\frac{|C|}{L(C)+O(|V|)}$. Since $L(C)$ is several times (≥ 3) the value of $\mid C \mid$, our algorithm requires a minimal partitioning of the buffer pool. Also, our algorithms impose no restrictions on routing and would not discard any packets. Livelock-freedom can also be easily added by taking into account the time of a packet's arrival at a node. Finally, because of the simplicity of the packet admission procedure at every node, our algorithms can handle packets more efficiently than the BS, FSET and FS algorithms [2,3].

References

[1] K. D. Günther. "Prevention of Deadlocks in Packet-Switched Data Transport Systems", *IEEE Trans. Commun., Vol. C-29, No. 4, Apr. 1981, pp. 512-524.*

[2] S. Toueg and J. D. Ullman. "Deadlock-Free Packet Switching Networks", *SIAM J. Comput., Vol. 10, No. 3, August 1981, pp. 594-611.*

[3] S. Toueg. "Deadlock- and Livelock-Free Packet Switching Networks", *Proc. 12th ACM Symposium on Theory of Computing, Los Angeles, California, Apr. 1980, pp. 94-99.*

[4] P. M. Merlin and P. J. Schweitzer. "Deadlock Avoidance in Store-and-Forward Networks–I: Store-and-Forward Deadlock", *IEEE Trans. Commun., Vol. C-28, No. 3. Mar. 1980, pp. 345-354.*

[5] I. S. Gopal. "Prevention of Store-and-Forward Deadlock in Computer Networks", *IEEE Trans. Commun., Vol. C-33, No. 12, Dec. 1985, pp. 1258-1264.*

[6] W. Wimmer. "Using Barrier Graphs for Deadlock Prevention in Communication Networks", *IEEE Trans. Commun., Vol. C-32, No. 8, Aug. 1984, pp. 897-901.*

[7] D. Gelernter. "A DAG-Based Algorithm for Prevention of Store-and-Forward Deadlock in Packet Networks", *IEEE Trans. Comput., Vol. C-30, No. 10, Oct. 1981, pp. 709-715.*

[8] J. Blacewicz, D. P. Bouvet, and G. Gambosi. "Deadlock-resistant flow control procedures for store-and-forward networks", *IEEE Trans. Commun., Vol. C-32, No. 8, Aug. 1984, pp. 884-887.*

[9] N.E. Gibbs. "Algorithm 491: Basic Cycle Generation", *CACM, Vol. 18, No. 5, May 1975, pp. 275-276.*

[10] N. Deo and M. Prabhu. "Algorithms for Generating Fundamental Cycles in a Graph", *ACM Trans. Math. Software, Vol. 8, No. 1, Mar. 1982, pp. 26-42.*

[11] N.E. Gibbs. "Algorithm 492: Generation of All the Cycles of a Graph from a Set of Basic Cycles", *CACM, Vol. 18, No. 6, Jun., 1975, pp. 310.*

[12] M.M. Syslo. "An Efficient Cycle Vector Space Algorithm for Listing All Cycles of a Planar Graph", *SIAM J. Comput., Vol. 10, No. 4, Nov., 1981, pp. 797-808.*

[13] C.L. Liu. *Introduction of Combinatorial Mathematics, McGraw-Hill, p. 193.*

Figure 1 – When Two Cycles Meet

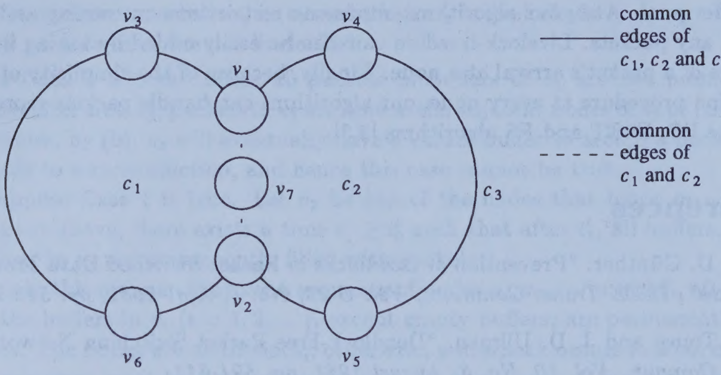

v_3
v_4
v_1
v_7
v_2
v_6
v_5
c_1
c_2
c_3
common edges of c_1, c_2 and c_3
common edges of c_1 and c_2

Figure 2 – Empty Buffer Exchange

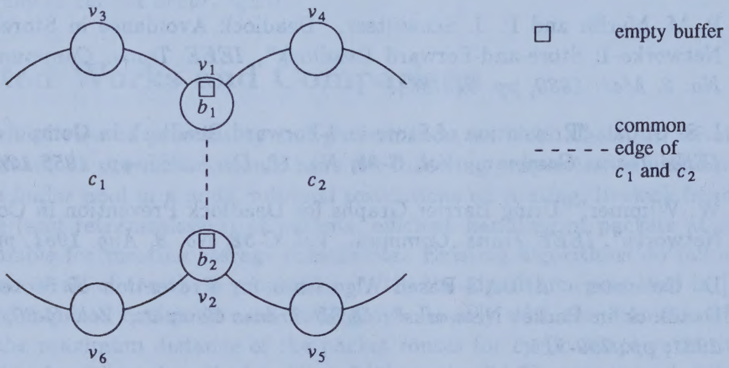
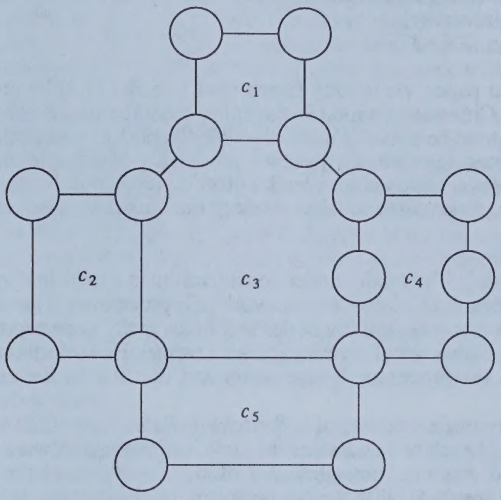

v_3
v_4
v_1
b_1
b_2
v_2
v_6
v_5
c_1
c_2
□ empty buffer
common edge of c_1 and c_2

Figure 3 – A Simple Complete Example



The set of cycles = {1, 2, 3, 4, 5, 13, 23, 35, 34, 132, 134, 135, 234, 235, 354, 1324, 1325, 1345, 2345, 12345}, where digit 1 corresponds to c_1, and so on.

Initially $C_1 = \{1, 2, 3, 4, 5\}$.
Step 1: checks 34 and finds that it is not a 2-c ring-cover. Now, $C_1 = \{1, 2, 3, 4, 5, 34\}$, and $C_2 = \{235, 1325, 2345, 12345\}$.
Step 2: $C_1 = C_1 \cup \{235\}$, $C_2 = \{2345, 12345\}$
Step 3 $C_1 = \{1, 2, 3, 4, 5, 34, 235, 2345\}$, $C_2 = \phi$.

Hence, the set of selected cycles = $\{1, 2, 3, 4, 5, 34, 235, 2345\}$.

A New Model and Fast Election in Fiber-optic Rings

Hosame Abu-Amara and Vidya Gummadi
Electrical Engineering Department,
Texas A&M University,
College Station, TX - 77843.

In our paper, we modify Cidon et al.'s model [1,3] by considering *queuing delays* in the nodes. In Cidon et al.'s model, the *entire* message queue can be processed in one time unit which might not be practically feasible.We consider a more realistic model where we take the queuing of messages into account, the processing of a single message requiring one time unit. We improve on Cidon et al.'s leader election algorithm by presenting an algorithm that requires *O(log n)* time units and takes message queuing delays into account.

THE MODEL:

Fiber-Optic Model : The model under consideration is a modified version of the novel model proposed by Cidon et al. Consider a network of *n* processors. The communication network is represented by a graph containing nodes and links, every node representing a processing unit and every link a bidirectional communication channel. Henceforth, we consider the node to be synonymous to the processor it represents and the link to the communication channel it represents.

Every node consists of a *Switching Subsystem* (SS) and a *Node Control Unit* (NCU). The SS of each node contains the communication hardware responsible for receiving, forwarding, and sending messages. All of the functions of the SS are implemented in hardware. As a result of this, the SS performs its duties rapidly, but due to the hardware implementation it cannot make decisions that are based on the information content of the messages. When the SS of a node *v* receives a message *M*, the subsystem sends a copy of *M* to *v*' s NCU, and depending on the header of *M*, either discards *M* or forwards *M* to another node.

The NCU of each node *v* contains the processing hardware and software necessary for queueing the messages, extracting the information content of messages, doing some internal computations, and generating messages to be forwarded to other nodes via *v*'s SS . Since the functions of the NCU are implemented in software, the NCU is slow and requires the presence of message queues to buffer the messages forwarded by the switching subsystem.

As in Cidon et al's model, we assume that when a node *u* wishes to send a message *M* to node *v*, then *u* encodes the entire route that *M* must follow in *M*'s header. When *u*'s SS receives *M*, the SS examines the header and places the message on the appropriate links. When *v*'s SS receives *M*, the subsystem forwards *M* to *v*'s NCU. A desirable feature is that there is an option for every node on the route of *M* to receive a copy of *M*.

Each node *v* has a *Send-Buffer(v)* and a *Receive-Buffer(v)* to store messages. Each processor in the network has a local clock on which it runs. All the processors run on a synchronized clock . Because fiber-optic networks have very large bandwidths, the number of messages used by an algorithm is not as important as the time complexity. Let M_v be a message acquired by processor *v*, and let M_{vr} be the message that *v* generates in response to *M*. A time unit in our model is defined to be

[Time required for NCU of *v* to remove M_v from the head of the message queue]
+ [Time required for NCU to process M_v and place M_{vr} in Send-Buffer(v)]

When the SS of *v* realizes that message M_v is destined for *v*, it forwards $M_v$to the message queue in the NCU of v. The NCU removes the message off the head of the message queue and processes it. After forwarding M_{vr} to the SS, the NCU looks at the queue for additional messages, removes the messages from the head of the queue one at a time and processes them. Since we assume that the processing of a single message requires one time unit, the time taken for processing a message queue of length *k* is *k* time units. In Cidon et al's model the entire message queue can be processed in one time unit which might not be practically feasible.

As in Cidon et al's model, we assume that link delays are negligible when compared with processing delays. This assumption holds only if each message does not traverse more than n links.

THE ALGORITHM

Overview : In the leader election problem, all the nodes start in the same state. Each node broadcasts its personal identification number upto a fixed depth with iterative deepening. At every node, a comparison of the *ID* s is made as a result of which some nodes might *kill* themselves. When we say that a node *kills* itself, we mean that it no longer acquires or generates messages. The SS of the dead nodes remain active and help in message-routing. Eventually, when the algorithm terminates there is exactly one live node which is designated as the leader.

A *round* as used in the algorithm consists of two *sub-rounds*. In the first subround, every node broadcasts its personal *ID* on its links. When a node receives messages on its links, it compares its *ID* with the IDs received. If the node finds any of the received IDs greater than its own, it rebroadcasts on its links the highest ID it has seen and kills itself. This rebroadcasting of the highest ID seen, constitutes the second subround. Since messages are broadcasted and received on all links incident on a node, the size of the message queue should also be considered because it contributes a significant amount to the time-complexity of the algorithm.

Details; : Assume *u* to be the node with the highest identification number *ID(u)* .

Round 1 : Every node *v* broadcasts *ID(v)* upto a distance of one link away from itself to its neighbors x and y . Similarly, node *v* receives messages *ID(x)* and *ID(y)* from its neighbors. This constitutes the first sub-round. Thus, *v* makes a comparison of *ID(v)* with *ID(x)* and *ID(y)* . We have the following cases:

(i) *ID(v) <ID(x)* or *ID(v) < ID(y)* : In the second subround,*v* rebroadcasts on its links the largest ID it has seen namely *MAX(ID(x),ID(y))* . Next node*v* kills itself, i.e. no longer acquires or generates any messages. It just routes the messages.

(ii) *ID(v) >ID(x)* and *ID(v) >ID(y)*: In this case *v* goes into the second sub-round. When *x* and *y* receive *ID(v)*, then *x* and *y* die and rebroadcast in the second subround the highest ID each one of them has seen. In the second sub-round *v* compares *ID(v)* with the highest IDs seen by *x* and *y*. If *v* is still alive after the second sub-round, then *v* proceeds to *Round 2* in the algorithm; else *v* stops executing the algorithm - it no longer generates any messages.

Round i (i => 2) : In the first sub-round of round *i,* every live node *k* sends *ID(k)* to a distance $d = 2^i$ from itself. Then every node receives messages on its links. (It is to be noted that when a live node sends a message *M* upto a distance $d = 2^i$ every node through which the message has been routed receives a copy of *M*.) Node *k* compares *ID(k)* with the IDs *k* received in round *i*. We have the following cases:

(i)*ID(k)* is not the largest identifier that *k* has seen: When *k* notices an ID greater than *ID(k)*, *k* kills itself. Before *k* kills itself, *k* sends the largest ID it has seen on its links upto a distance $d = 2^i$ · Node *k* no longer acquires or generates any messages.

(ii) *ID(k)* is the largest identifier that *k* has seen: In this case *k* goes into the second subround and every node *v* that had sent a message to *k* dies when *v* notices *ID(k)*. When *v* dies, *v* sends in the second sub-round to a distance *d* the largest ID that *v* has seen. In the second sub-round, *k* compares *ID(k)* with the highest ID seen by every node *v*. After the second sub-round, if *k* is live, then it moves on to the next round; else it stops executing the algorithm. After *log n* rounds there will be a unique live node that is declared the leader. In the case where *n* , the total number of nodes in the network is not known, once the node overshoots itself (it gets its own message) it declares itself to be the leader and the execution of the algorithm is terminated. □

Recently, we extended this algorithm to all synchronous networks in which the nodes know the network topology. The algorithm runs in O($n^{3/4}$ log log n + r) time units where *r* is the degree of the network.

REFERENCES

[1] I.Cidon, I.Gopal, and S.Kutten, " New models and algorithms for future networks", manuscript; *Proc. 7th ACM Symp. Principles Distributed Comput.*, Canada, Aug. 1988.

[2] I.Cidon and I.Gopal, " *PARIS* : An approach to private integrated networks", *Journal of analog and Digital Cabled Systems*, to appear.

[3] Hosame Abu-Amara, " A fast topology maintainence algorithm for high-bandwidth networks",*Proc. 27th Annual Allerton Conference on Communication, Control, and Computing* , Monticello, Illinois, Sep. 1989.

ADAPTIVE ROUTING FOR CONGESTION CONTROL IN MULTIPATH REGULAR NETWORKS

RAM KRISHNAN & J. A. SILVESTER

Electrical Engineering - Systems Department
Communication Sciences Institute
University of Southern California
Los Angeles, CA 90089-0781

ABSTRACT

This paper studies the application of the Join-Biased Queue rule to reduce network congestion by achieving optimal flows in regularly structured networks. The rule makes use of "free choice routing decisions" when multiple paths with the same number of hops exist. We have chosen ShuffleNet in this paper though the above approach can be applied to other networks with regular topologies like the Manhattan Street network.

1. INTRODUCTION

It is our belief that, in the future, high speed networks will employ regular topologies possessing multiple minimum hop paths for their logical framework. One recently proposed lightwave network uses ShuffleNet as the logical topology [1],[2]. The Manhattan Street Network (MS-Net) is another regular structured topology proposed for use in Metropolitan Area Networks (MANs) [3]. These networks belong to a class most commonly referred to as multipath regular networks.

Congestion Control in high speed networks is still an open area of research as traditional solutions for network congestion, like feedback control, are not very effective due to the large number of messages in transit in the optical fiber pipe. One way to reduce congestion in these networks is to adopt path splitting; if the optimal path is congested, the message is adaptively routed, depending on the load, along alternate optimal paths. Throughout this paper, we refer to minimum hop paths as optimal paths between source-destination pairs. These adaptive routing schemes should be simple to implement and should use only local information for their decisions. We discuss the different types of adaptive routing strategies currently in use and make a case for the Join-Biased Queue rule. In short, we propose that adaptive routing be used as a framework for reducing network congestion.

In section 2, we provide a definition of multipath regular networks, presenting a few typical examples. In particular, we descibe the topology of the ShuffleNet and the MS-Net. We discuss the main features of adaptive routing algorithms for these networks and describe our proposed scheme in detail in section 3. The Join-Biased Queue (JBQ) rule is discussed in section 4 wherein we show how to determine the queue length distribution by solving a Markov chain numerically. In section 5 we study the performance of the JBQ adaptive routing model through some examples. Finally, we conclude the paper in section 6.

2. MULTIPATH REGULAR NETWORKS

Metropolitan Area Networks are increasingly employing multipath regular topologies for their logical configurations. A key feature of these networks is that there exists multiple minimum-hop paths between source-destination pairs. As a result, adaptive routing schemes, which make use of these multipaths, can be employed to bring about a more balanced loading on the channels in the network and hence reduce network congestion.

A variety of networks fall into this category. In this paper, we consider ShuffleNet[2] as a representative of this class of networks and study the performance improvements obtained by adaptive routing. To construct the ShuffleNet connectivity graph, we arrange $N = kp^k$ $(k = 1, 2, \cdots; p = 1, 2, \cdots)$ users in k columns of p^k users each. Moving from left to right, successive columns are connected by p^{k+1} directed arcs, arranged in a fixed shuffle pattern, with the last column connected to the first as if the entire graph were wrapped around a cylinder. Each of the p^k users in a column has p arcs directed to p different users in the next column. As an example, figure 1 illustrates an eight user ShuffleNet with each user having two fixed-wavelength transmitters and two fixed-wavelength receivers (i.e., $k = 2$, $p = 2$). Throughout this paper, we use the number of hops as the distance parameter.

For a particular source-destination matrix, two different types of message streams can be identified. First, there are those messages that have a unique minimum hop path to their destination which we refer to as TYPE S messages [6]. Second, there are TYPE M messages that have multiple minimum hop paths to their destination. For example, in figure 1, there are two minimum-hop routing paths (of length 3) from node 0 to node 6: $0 \to 4 \to 1 \to 6$, $0 \to 5 \to 3 \to 6$. However, from node 0 to node 3 there is only one minimum-hop routing path (of length 2): $0 \to 5 \to 3$. If a message cannot reach its destination in one pass around the network, (i.e., k hops) and the minimum-hop routing path length is $k + D$, say, then the routing is arbitrary in the first D hops. In other words, a TYPE M message can be routed arbitrarily for the first D hops after which a single path of length k leads to the destination. With reference to the eight node example where $k = 2$, a message from 0 to 6 has to travel at least 3 hops which implies that it can be routed arbitrarily for the first hop.

The Manhattan Street Net (MS-Net) [4] is another example of a network with regular topology. The nodes in the MS-Net, as illustrated in figure 2, are connected in a grid with adjacent rows and columns traveling in opposite directions. The wraparound links have the effect of logically placing the grid on the surface of a torus so that there are no corners. There are many nodes at which selection of either link results in the same number of hops to the destination. For example, there are two minimum hop paths of length 4 from node (1,4) to node (3,2): $(1,4) \to (2,4) \to (3,4) \to (3,3) \to (3,2)$, $(1,4) \to (1,3) \to (1,2) \to (2,2) \to (3,2)$. Therefore we can identify TYPE S and TYPE M messages in MS-Net just like in ShuffleNet.

A TYPE S message is normally routed according to the fixed (single path) routing scheme so that it follows the minimum-hop path to its destination. But if the desired output buffer is full, the packet is bumped to the smallest output queue. In other words, the message is deflected from its shortest path. In this paper, we have not incorporated the impact of deflection routing. Since a TYPE M message has no preference which output channel it goes to, it can be routed based on a simple adaptive routing scheme. In the next section, we describe two such schemes.

3. ADAPTIVE ROUTING SCHEMES

Under uniform traffic conditions, ShuffleNet makes efficient use of the channels with a simple fixed (single path) routing scheme [5]. However, this provides a balanced loading of the channels only when the traffic matrix is uniform. Under certain non-uniform traffic patterns, fixed routing algorithms cause a large number of messages to pass through a small subset of channels and overload them. In practice, a uniform traffic matrix is highly unlikely, and a traffic pattern where most of the messages are generated by a subset of nodes is closer to reality.

Static routing schemes attempt to provide a balanced loading under non-uniform traffic patterns using probabilistic splitting. Adaptive routing algorithms provide a more balanced utilisation of the channels by exploiting the existence of multiple minimum hop paths. We study and compare two such adaptive routing schemes that not only relieve the traffic congestion resulting from nonuniform traffic patterns and failures of transmission channels and nodes, but also the bottlenecks caused by random packet generation. Karol and Shaikh proposed a simple, distributed, adaptive routing scheme and demonstrated its ability to support nonuniform traffic patterns and reduce the mean queue size and variance through simulation [6]. The first scheme we consider here is similar except that we do not resort to bumping messages that have a preferred routing path. In this scheme, when a node receives a message that can be routed arbitrarily to any of its output channels without affecting the path length to the message's destination, it attempts to minimize network congestion by routing the message to the output port with the fewest messages queued for transmission. This scheme uses only local queue size information at a node, and is similar to the Join Shortest Queue (JSQ) rule. Moreover, the algorithm adapts automatically to time-varying conditions and does not require apriori knowledge of the traffic patterns.

The second scheme uses the Join Biased Queue rule - a generalisation of the JSQ rule as the basis for the adaptive routing strategy. This permits a more balanced bifurcation of the adaptive traffic that can be routed arbitrarily to any of its output channels without affecting the path length to the message's destination. This approach exhibits improvement over the first scheme in terms of channel loading, mean queue size, variance and blocking probability. The approach uses the Join Biased Queue-Best Stochastic (JBQ-BS) rule proposed by T. S. Yum and M. Schwartz [7], wherein local JBQ adaptivity is superimposed on the fixed flow deviation (BS) rule base. The flow deviation algorithm[8] specifies the flow deviation in the network and the JBQ rule, with its inherent bifurcation ability, regulates the traffic so as to achieve that flow distribution. In other words, we want the same link utilisations for these two rules.

For a given source-destination traffic matrix, the flow deviation algorithm is invoked to compute the flows on all the links in the multihop network wherein the objective is to minimize the overall mean delay for a message to traverse the network. The flow pattern obtained in this manner is the optimum pattern for stochastic fixed rules and serves as a satisfactory heuristic for computing the flows in the network. Using these flows, the fixed arrival rates γ_1 and γ_2 (for the two outgoing queues) and the adaptive arrival rate γ are identified for each node, where traffic has to be bifurcated to achieve a balanced channel loading. The JBQ rule is then employed to split the adaptive traffic such that the utilisation of each queue is the same as that given by the flow deviation algorithm. Then the Markov chain of the JBQ rule is solved repeatedly with different values for the parameters $(\triangle, \beta)$defined in detail in the next section. Each set of solutions gives a set of link utilisations (ρ_1, ρ_2). The needed set of $(\triangle, \beta)$ is that which gives (ρ_1, ρ_2) closest to the values predicted by flow deviation. The numerical solution of the Markov chain affords the computation of the steady state occupancy probabilities of both the queues and hence, their mean queue sizes and variances. Once, a $(\triangle,\beta)$ pair is determined for a particular node, the adaptive routing algorithm uses the information to route the adaptive traffic in accordance to the JBQ rule. It is a distributed routing scheme that bases routing decisions on information contained in an arriving message's header and the local queue sizes.

This scheme is particularly easy to implement in regular topologies. The key is to determine $(\triangle,\beta)$ for each node. For a node with two incoming links and two outgoing links a simple search procedure can be used. Exhaustive search becomes prohibitively complex for a node with three or more outgoing links. For the three link case, we have a three queue system with fixed messages with rates γ_1, γ_2 and γ_3 joining their respective queues and an adaptive stream at a rate γ to be split into three streams of rates a_1, a_2 and a_3. The problem is how to set the bias level on each queue so as to achieve the specified flow distribution. A three dimensional Markov chain has to be solved for the three queue system and the states have to be partitioned into three regions for the three routing decisions; the incoming message should join Q_1, Q_2 and Q_3. Some simple approximation methods have been investigated wherein we have used decoupled iterative techniques. The question of setting the bias level on each queue still remains and we suspect that even though the flows are complex functions of the thresholds (i.e., bias), a simple search procedure should prove sufficient.

The primary drawback of this scheme is that the traffic pattern has to be known apriori and the algorithm cannot really adapt automatically to rapidly time-varying conditions. For a given traffic pattern, the algorithm has to be invoked to compute the flows in the network and compute the bias for each node trying to adaptively bifurcate the traffic. In the next section, we discuss the JBQ rule concept in detail.

4. THE JBQ RULE CONCEPT

In this section, we study the JBQ rule in the context of a node with two transmitters and two receivers. Its extension to the ShuffleNet with the incorporation of the flow deviation rule will be studied in the next section.

An isolated node and its queueing model is considered in figure 3. Let γ_1, γ_2 represent Poisson arrival rates of messages that are constrained to be sent out via link 1 and link 2, respectively. They are called fixed arrival rates. Let q_1, q_2 be the lengths of queue 1 (Q_1) and queue 2 (Q_2); and let λ_1, λ_2 be the actual input rates to Q_1 and Q_2. Let all queues be formed in finite buffers of size M and let γ represent the adaptive Poisson arrival rate of messages that can be switched to either Q_1 and Q_2.

In words, the JBQ rule says: route a message in the adaptive category to Q_1 if the length of Q_1 is less than the length of Q_2 plus $\triangle$; route it to Q_2 if greater. If the length of Q_1 equals the length of Q_2 plus $\triangle$, then route it to Q_1 and Q_2 with probability β and $(1-\beta)$ respectively. The special case $(\triangle,\beta) = (0, 0.5)$, $\gamma_1 = \gamma_2 = 0$, reduces to the JSQ rule. The following is the JBQ rule to route the γ messages:

$$\left.\begin{aligned}\lambda(q_1) &= \gamma_1 + \gamma \\ \lambda(q_2) &= \gamma_2\end{aligned}\right\} \quad \text{when } q_1 < q_2 + \triangle$$

$$\left.\begin{aligned}\lambda(q_1) &= \gamma_1 \\ \lambda(q_2) &= \gamma_2 + \gamma\end{aligned}\right\} \quad \text{when } q_1 > q_2 + \triangle$$

$$\left.\begin{aligned}\lambda(q_1) &= \gamma_1 + \beta\gamma \\ \lambda(q_2) &= \gamma_2 + (1-\beta)\gamma\end{aligned}\right\} \text{ when } q_1 = q_2 + \triangle$$

Here, $\triangle$ in $\{0,\pm1,\pm2,\cdots\}$ is an integer representing the bias level and β in $[0,1)$ is the *apriori* probability for routing the γ messages to Q_1 when $q_1 = q_2 + \triangle$. Thus, β is a fine tuning parameter.

For a particular set of input parameters $\{\gamma,\gamma_1,\gamma_2,\triangle,\beta,M\}$, the JBQ rule can be represented by a two-dimensional Markov chain with the transition rates dependent on the state (q_1, q_2). This is possible because Poisson arrival rates and exponential service rates are assumed. Let $P_{i,j} = \text{Prob}\ [q_1 = i, q_2 = j]$. The complete set of states for $\triangle = 2$ is shown in figure 4. Groups of states with the same inward-outward transition rates can be identified from the sketch, with the state equations for these states differing from each other only by the indices i and j. A typical set, for example, the one corresponding to region 3, may be written as follows (the average message length is normalized to unity):

$$P_{0,j} = \frac{\gamma_2 P_{0,j-1} + P_{1,j} + P_{0,j+1}}{1+\gamma_1+\gamma_2+\gamma} \qquad j = 1,2,\cdots,M-1. \tag{1}$$

These equations are then numerically solved to find $P_{i,j}$. These provide information as to the mean queue sizes and variances and blocking probabilities of both the queues. In the next section, we consider two examples of non-uniform traffic patterns and show how adaptive routing schemes distribute the loading evenly on all the links in the network and thereby reduce network congestion.

5. PERFORMANCE OF THE ADAPTIVE ROUTING MODEL

The interest in adaptive routing is primarily for handling non-uniform and time-varying traffic patterns. The performance of the JBQ approach has been studied for the eight-NIU ($p = 2, k = 2$) ShuffleNet with a non-uniform traffic matrix as shown in the figure 5. A buffer size of ten was assumed in all our calculations. The external traffic rates assumed are as indicated in Table 1.

Table 1

Commodity k	Source-Destination pair	Possible minimum hop routes	Rate
1	$0 \to 6$	$0 \to 4 \to 1 \to 6$ $0 \to 5 \to 3 \to 6$	λ
2	$1 \to 5$	$1 \to 6 \to 0 \to 5$ $1 \to 7 \to 2 \to 5$	λ
3	$2 \to 1$	$2 \to 4 \to 1$	λ
4	$3 \to 2$	$3 \to 7 \to 2$	λ
5	$4 \to 3$	$4 \to 0 \to 5 \to 3$ $4 \to 1 \to 7 \to 3$	λ
6	$5 \to 4$	$5 \to 2 \to 4$	λ
7	$6 \to 7$	$6 \to 1 \to 7$	λ
8	$7 \to 0$	$7 \to 2 \to 4 \to 0$ $7 \to 3 \to 6 \to 0$	λ

The total average delay is written as

$$T = \frac{1}{8\lambda}\sum_{i=1}^{16}\frac{f_i}{1-f_i} \tag{2}$$

The flows f_i that minimize T, the overall average delay, are determined.

Table 2

Rules	$f_{0\to4}$	$f_{0\to5}$	$f_{1\to6}$	$f_{1\to7}$	$f_{4\to0}$	$f_{4\to1}$
Flow-deviation (BS)	0.2339	0.4821	0.3504	0.4840	0.2997	0.5347
JBQ-BS	0.2336	0.4824	0.3503	0.4841	0.3500	0.5344
$(\triangle,\beta)$ that achieves this bifurcation	(0,0.775)		(0,0.08)		(2, 0.99)	

The first row of Table 2 shows the flows derived from applying the flow-deviation algorithm. The second row shows the best flows found using the JBQ rule and the last row lists the $(\triangle,\beta)$ values used to achieve them. Table 3 compares the link flows obtained with fixed routing and with adaptive routing using the JBQ rule. Clearly, adaptive routing permits a more balanced distribution of the traffic in the network and it is also optimal in the sense that the overall average delay is minimized.

Table 3.

Routing	$f_{0\to4}$	$f_{0\to5}$	$f_{1\to6}$	$f_{1\to7}$	$f_{2\to4}$	$f_{2\to5}$	$f_{3\to6}$	$f_{3\to7}$
Fixed Routing	0.3	0	0.3	0.9	0.9	0.3	0	0.3
Adaptive Routing (JBQ)	0.23	0.48	0.35	0.49	0.6	0.18	0.36	0.3

Fixed routing forces four channels to carry messages for as many as three source-destination pairs, so the network saturates at $\lambda \leq 0.33$. For fixed routing and $\lambda = 0.3$, the maximum (over all NIUs) mean queue size is 3.45 messages with a variance of 10.2. The JBQ rule gives a maximum mean queue size of 1.50 and a maximum variance of 3.76. While the employment of the JBQ rule yields a maximum load of 0.6, Karol and Shaikh [6], using simulations, report a maximum load of 0.76 using the JSQ rule in conjunction with deflection routing. The blocking probabilities at all the nodes were determined to be negligible showing that the buffers were empty all the time for $\lambda = 0.3$. The results are summarized in Table 4.

Table 4

Traffic Pattern	Routing and Loading	Max. Load	Maximim Mean Q Size	Maximum Q Size Var.
Fig 5.	Fixed Routing, $\lambda = 0.3$	0.90	3.45	10.2
	JSQ with defl. Routing, $\lambda = 0.3$	0.76	1.55	2.10
	JBQ-BS Routing, $\lambda = 0.3$	0.60	1.50	3.76

As a second example, the performance of the model has been studied for the traffic pattern shown in figure 6.

Fixed routing schemes fare very poorly with the network saturating at λ =0.19, the link from node 0 to node 1 being the bottleneck. The JSQ rule with deflection routing [6] performs better, saturating only at $\lambda = 0.45$. The algorithm we have proposed exhibits even superior performance allowing an arrival rate as high as 0.6. The mean queue sizes and variances are also much lower than the other two rules. The results are summarized in Table 5.

Table 5

Traffic Pattern	Routing and Loading	Max. Load	Maximum Mean Q Size	Maximum Q Size Var.
Fig 6.	Fixed Routing, $\lambda = 0.19$	0.95	8.03	63
	JSQ with defl. Routing, $\lambda = 0.19$	0.75	1.28	1.36
	JBQ-BS Routing, $\lambda = 0.19$	0.33	0.39	0.62

6. CONCLUSIONS

We have proposed path splitting, wherein traffic is adaptively routed along alternate minimum hop paths, as a means of reducing congestion in high speed networks. The approach suggested in this paper is particularly relevant in multipath regular networks, possessing multiple minimum hop paths between source-destination pairs. Specifically, we have studied the improvement gained by adopting the adaptive routing algorithm in the ShuffleNet and the Manhattan Street Net. The algorithm we propose uses only local queue size information and thus, does not involve a lot of message processing at the intermediate switching nodes.

The adaptive routing scheme considered here, uses *apriori* information about the traffic patterns to distribute the loads more evenly on the channel and thereby, reduce congestion resulting from non-uniform traffic patterns. In order to obtain the mean sample queue sizes, variances and blocking probabilities from an analytical approach, the Join Biased Queue rule was superposed on the flow deviation algorithm. We considered two examples of non-uniform traffic patterns and the results obtained exhibited superior performance in terms of channel loading and blocking probabilities than the adaptive routing schemes proposed in [6].

References

[1] A. S. Acampora, "A Multichannel Multihop Local Lightwave Network," *GLOBECOM' 87 Conf. Rec.*, pp. 1459-1467, Nov. 1987.

[2] M. G. Hluchyj and M. J. Karol, "ShuffleNet: An Application of Generalized Perfect Shuffles to Multihop Lightwave Networks," *INFOCOM' 88 Conf. Proc.*, pp. 379-390, Mar. 1988.

[3] A. S. Acampora and Mark. J. Karol, " An Overview of Lightwave Packet Networks, " *IEEE Network*, pp. 29-41, Jan. 1989.

[4] N. F. Maxemchuck, " Regular and Mesh Topologies in Local and Metropolitan Area Networks, " *AT&T Tech. Jour.*, pp. 1659-1686, Sept.1985.

[5] M. Eisenberg and N. Mehravari, " Performance of the multichannel multihop lightwave network under nonuniform traffic," *IEEE Journal on Selected Areas in Communications*, pp. 1063-1078, Aug. 1988.

[6] Mark. J. Karol and Salman Shaikh, "A Simple Adaptive Routing Scheme for ShuffleNet Multihop Lightwave Networks," *GLOBECOM' 88 Conf. Proc.*, pp. 1640-1647, Nov. 88.

[7] Tak-Shing. P. Yum and Mischa Schwartz, "The Join-Biased-Queue Rule and Its Application to Routing in Computer Communication Networks," *IEEE Trans. Commun.*, vol. COM-29, pp. 505-511, April 1981

[8] D. Bertsekas and R. Gallagher, "Data Networks," *Prentice Hall International, New York* 1987.

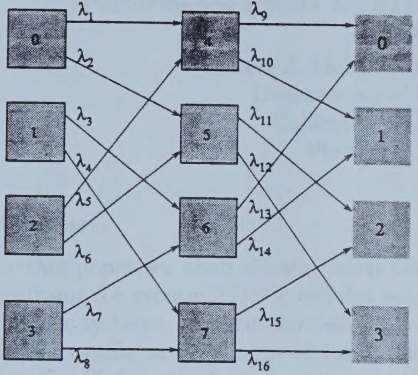

λ1
λ2
λ3
λ4
λ5
λ6
λ7
λ8
λ9
λ10
λ11
λ12
λ13
λ14
λ15
λ16
0
1
2
3
4
5
6
7
0
1
2
3

Figure 1. 8-NIU (p = 2, k =2) ShuffleNet Connectivity Graph

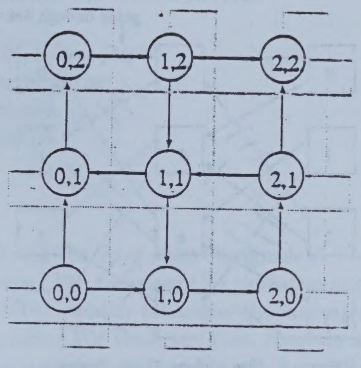

0,2
1,2
2,2
0,1
1,1
2,1
0,0
1,0
2,0

Figure 2. Topology of the MS-Net

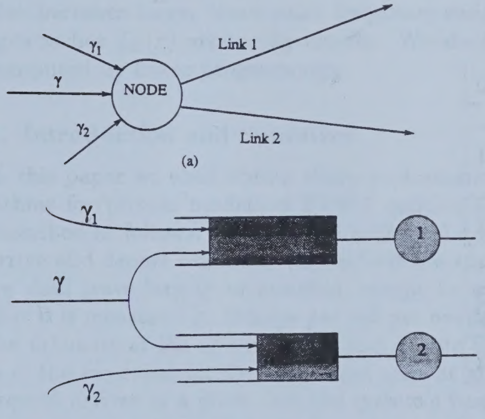

γ1
γ
γ2
NODE
Link 1
Link 2
(a)
γ1
γ
γ2
1
2

Figure 3. An isolated network node and its queueing model.
(a) An isolated network node. (b) Queueing model.

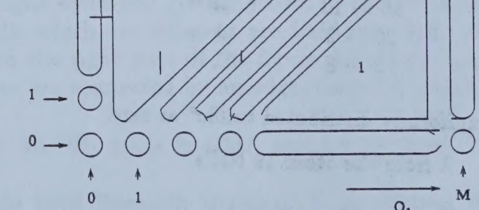

M
Q2
3
2
1
1
0
0
1
M
Q1

Figure 4. Grouping of states for the defining equation, JBQ rule, Δ = 2.

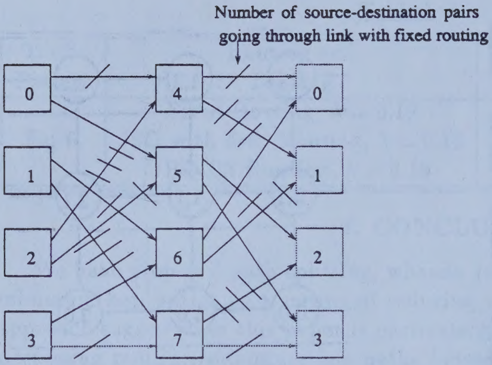
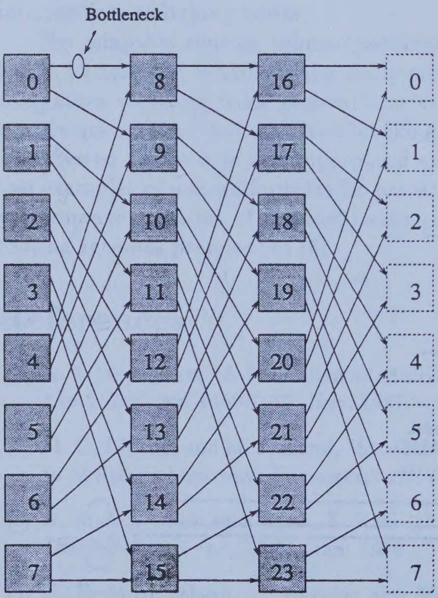

Number of source-destination pairs going through link with fixed routing
0
1
2
3
4
5
6
7
0
1
2
3

Figure 5. Non uniform Traffic Pattern


Bottleneck
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
0
1
2
3
4
5
6
7

Source-Destination	Rate
0 → 20	λ
1 → 16	λ
16 → 12	λ
20 → 2	λ
23 → 8	λ

plus "uniformly distributed traffic" at rate

λ from the other 19 NIUs

Figure 6. Nonuniform Traffic Pattern.

Performance Limits for FDMA Cellular Telephone Systems*

Robert J. McEliece and Kumar N. Sivarajan**
Department of Electrical Engineering
California Institute of Technology
Pasadena, California 91125

Abstract.

In this paper we shall obtain sharp performance limits for frequency-assignment algorithms for certain FDMA cellular telephone systems. We measure the performance of such systems by their *carried traffic functions*, $T(r)$, which measure the average carried traffic as a function of the average offered traffic. We shall see that there is a function $T_G(r)$, which depends only on the underlying system topology G (a finite undirected graph), such that for any frequency-assignment algorithm, $T(r) \leq T_G(r)$. On the other hand, we shall show that asymptotically as the number of available frequencies becomes large, there exist frequency-assignment algorithms whose performance approaches $T_G(r)$ arbitrarily closely. We shall show that the function $T_G(r)$ can be computed by linear programming.

1. Introduction and Summary.

In this paper we shall obtain sharp performance limits for frequency-assignment algorithms for certain models of FDMA cellular telephone systems. Our models can be described as follows. There are N cells, and n available user frequencies. Call requests arrive and depart randomly throughout the system, according to a probabilistic model we shall leave largely unspecified, except to say that it is the same for all cells, and that it is measured in *Erlangs per cell per available frequency*, a parameter we shall call the *intensity of the offered traffic* and denote by r. Thus if the offered traffic intensity is r, the expected number of offered calls at any time is rn in each cell. When a call request arrives in a given cell, the system's frequency-assignment algorithm can either honor the request or block it. Honoring a call request means assiging it to one of the n user frequencies; blocking a call means ignoring it completely. Calls in progress do not move between cells; thus we do not consider call "handoffs." If a call request in cell i is honored, the frequency assigned to it must satisfy certain constraints. The frequency must not already be in use in cell i. Nor must it be in use in any "nearby" cell. In this paper, the notion of "nearby" is binary and symmetric: for every pair of cells, that pair is either classified as nearby or not. It follows that the topology of our systems can be modelled by finite undirected graphs with N vertices corresponding to the N cells; two vertices are connected by an edge if and only if the corresponding cells are forbidden to use the same frequency simultaneously.

For example, consider the simple seven-cell system shown in the left part of Figure 1. If we assume that two cells which are adjacent are forbidden from using the same frequency, then the graph in the right part of the figure describes this system, it being understood that if vertices are connected by an edge, then the corresponding

---

* This work was supported by AFOSR grant 88-0247, and grants from GTE and Pacific Bell.

** Kumar Sivarajan is now with IBM Research Division, T. J. Watson Research Center, Yorktown Heights, New York 10598.

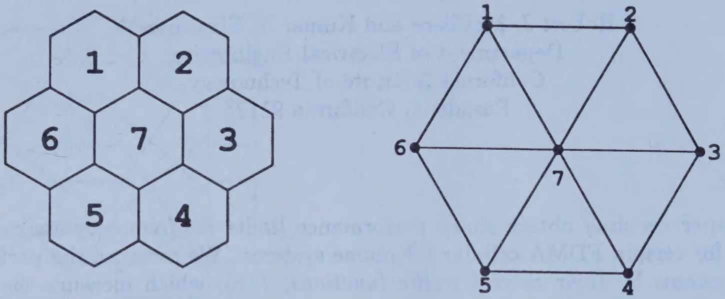


Figure 1. A seven cell system and the corresponding graph.

cells cannot use the same frequency. If G is an arbitrary graph, we shall call the cellular system whose topology is modelled by G the *G-system.*

The natural measure of the performance of a given frequency-assignment algorithm is its *blocking probability* as a function of the intensity of the offered traffic, a function we denote by $P_{\text{block}}(r)$. We have found it easier, however, to work with an equivalent performance measure which we call the *carried traffic function* $T(r)$, defined as

$$T(r) = r(1 - P_{\text{block}}(r)). \tag{1.1}$$

This function measures the expected number of calls being serviced by the system as a function of the offered traffic intensity. A study of the function $T(r)$ is the central concern of this paper. For a given graph G, we shall define a function $T_G(r)$, which can be computed by linear programming, and which has the following significance. If $T(r)$ denotes the carried traffic function for any frequency-assignment algorithm for the G-system, then $T(r) \leq T_G(r)$. On the other hand, in the limit as the number of available frequencies becomes large, there exist frequency-assignment algorithms for the G-system whose carried traffic functions are arbitrarily close to $T_G(r)$. Thus $T_G(r)$ can fairly be called *the* carried traffic function for the G-system.

The organization of the paper is as follows. In Section 2, we present some preliminary material about graphs, including a discussion of what we call *random graph multicolorings*, a notion which is central to our analysis of frequency-assignment algorithms. In Section 3 we will show that for any frequency-assignment algorithm, the carried-traffic function must satisfy $T(r) \leq T_0(r)$, where $T_0(r)$ is a simple function that can be computed by linear programming. In Section 4, on the other hand, we will give an asymptotic analysis of a class of "fixed" frequency-assignment algorithms, and show that in the limit as $n \to \infty$, these algorithms achieve carried traffic functions that are at least as large as $T_1(r)$, another simple function that can be computed by linear programming. Finally in Section 5 we will show that $T_0(r) = T_1(r)$. This common value, denoted by $T_G(r)$, is the function referred to above.

2. Graph Multicolorings and Random Graph Multicolorings.

Let G be an undirected graph with N vertices $\{v_1, v_2, \ldots, v_N\}$. An *independent set* for G is a set of vertices such that no two are joined by an edge. A *maximal independent set* is an independent set which is not a proper subset of any other independent set. We assume G has S maximal independent sets $\{V_1, V_2, \ldots, V_S\}$. For future reference, we also define the *indicator set* I_j for the maximal independent set V_j as $I_j = \{i : v_i \in V_j\}$, and the incidence matrix $A = (a_{ij})$ as

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \in V_j \\ 0 & \text{if } v_i \notin V_j. \end{cases}$$

The graph G can be reconstructed from A, and for our purposes it is the preferred representation of G.

If $\mathbf{w} = (w_1, w_2, \ldots, w_N)$ is a list of numbers assigned to the vertices of G, define the *v–m transform* (vertex-maximal independent set) of $\mathbf{w}$ as $\mathbf{W} = (W_1, W_2, \ldots, W_S)$, where $\mathbf{W} = \mathbf{w}A$, i.e.,

$$W_j = \sum_{i:i\in I_j} w_i = \sum_{i=1}^{N} w_i a_{ij} \quad \text{for } j = 1, 2, \ldots S. \tag{2.1}$$

For example, if $\mathbf{w} = (1, 1, \ldots, 1)$, then $\mathbf{W} = (N_1, N_2, \ldots, N_S)$, where N_j is the size of the jth maximal independent set V_j. Similarly, if $\mathbf{X} = (X_1, X_2, \ldots, X_S)$ is a list of numbers assigned to the maximal independent sets of G, the *m-v transform* (maximal independent set-vertex) of $\mathbf{X}$ is $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, where $\mathbf{x}^T = A\mathbf{X}^T$, i.e.,

$$x_i = \sum_{j:i\in I_j} X_j = \sum_{j=1}^{S} X_j a_{ij} \quad \text{for } i = 1, 2, \ldots N. \tag{2.2}$$

For example, if $\mathbf{X} = (1, 1, \ldots, 1)$, then $\mathbf{x} = (S_1, S_2, \ldots, S_N)$, where S_i denotes the number of maximal independent sets containing vertex v_i.

For example, consider the graph of Figure 1. The maximal independent sets are

$$V_1 = \{1, 3, 5\} \quad V_2 = \{2, 4, 6\} \quad V_3 = \{7\},$$

and the matrix A is

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Note that the S_i's are the column sums of A, and the N_j's are the row sums of A, i.e., $(S_1, S_2, S_3) = (3, 3, 1)$ and $(N_1, N_2, \ldots, N_7) = (1, 1, 1, 1, 1, 1, 1)$.

An *n-multicoloring* of G is an assignment of a set of distinct elements ("colors") from $\{1, 2, \ldots, n\}$ to each vertex in such a way that if v_i and v_j are adjacent vertices, the sets of colors assigned to v_i and v_j are disjoint. An n-multicoloring can be described by an $N \times n$ matrix (m_{ic}) of 0's and 1's such that $m_{ic} = 1$ if color c is assigned to vertex i, and 0 otherwise. Given an n-multicoloring, if the number of colors assigned to v_i is m_i, then plainly

$$m_i = \sum_{c=1}^{n} m_{ic}. \tag{2.3}$$

Theorem 2.1. *Suppose (m_{ic}) is an n-multicoloring of the graph G. If $(w_1, w_2, \ldots, w_N)$ is any set of nonnegative weights attached to the vertices of G, we have*

$$\sum_{i=1}^{N} w_i m_i \leq n W_{\max}, \tag{2.4}$$

where $W_{\max}$ is the maximum component of the v–m transform of $(w_1, w_2, \ldots, w_N)$.

Proof: From (2.3) we have that

$$\begin{aligned} \sum_{i=1}^{N} w_i m_i &= \sum_{i=1}^{N} w_i \sum_{c=1}^{n} m_{ic} \\ &= \sum_{c=1}^{n} \sum_{i=1}^{N} w_i m_{ic}. \end{aligned} \tag{2.5}$$

For a fixed value of c, the inner sum in (2.5) is equal to $\sum_{i \in J_c} w_i$, where $J_c = \{i : m_{ic} = 1\}$. But by the definition of a multicoloring, the set of vertices assigned a fixed color must be an independent set, so that $J_c \subseteq I_j$ for some index j. Hence (recall that the weights w_i are nonnegative)

$$\sum_{i=1}^{N} w_i m_{ic} = \sum_{i \in J_c} w_i \leq \sum_{i \in I_j} w_i = W_j \leq W_{\max}. \tag{2.6}$$

Thus combining (2.5) and (2.6), we have

$$\sum_{i=1}^{N} w_i m_i \leq \sum_{c=1}^{n} W_{\max} = n W_{\max},$$

as asserted. ❐

We next define a *random n-multicoloring of G* as a random $N \times S$ matrix $M = (m_{ic})$ of 0's and 1's such that for each point ω in the underlying sample space, $M(\omega)$ is an n-multicoloring of G.

Theorem 2.2. *If M is a random n-multicoloring for G, then for any set of numbers $(y_1, y_2, \ldots, y_N)$ satisfying $0 \leq y_i \leq 1$, we have*

$$E\left(\sum_{i=1}^{N} m_i\right) \leq \sum_{i=1}^{N} E(m_i) y_i + n \max_j (N_j - Y_j),$$

where N_j is the size of the jth maximal independent set V_j, and $(Y_1, Y_2, \ldots, Y_S)$ is the v–m transform of $(y_1, y_2, \ldots, y_N)$.

Proof: For any ω we have

$$\sum_{i=1}^{N} m_i(\omega) = \sum_{i=1}^{N} m_i(\omega) y_i + \sum_{i=1}^{N} m_i(\omega)(1 - y_i).$$

The theorem now follows by applying Theorem 2.1 to the second sum on the right side, and then taking expectations of both sides. ❐

3. Upper Bounds on the Performance of Frequency-Assignment Algorithms.

We now assume that our graph G serves as a model for a FDMA cellular telephone exchange. (One of the earliest appearances of graph models for cellular systems is in [5].) There are n frequencies available and the vertices of G represent the cells of the system. Two adjacent vertices of G represent cells which are forbidden to use the same frequency simultaneously. We assume that calls arrive randomly, at the same rate in each cell, and that the normalized traffic intensity is r Erlangs per cell per available frequency, i.e., the expected number of offered calls in any cell is rn. We assume that arriving calls are either assigned a frequency, or blocked, according to a particular frequency assignment algorithm. We shall make no formal attempt to define a frequency-assignment algorithm except to assume that any such algorithm produces a random n-multicoloring of G, viz., at any point in time the frequencies being used in adjacent cells are disjoint. If we denote the number of frequencies being used in cell i by m_i, then the expected value of the carried traffic is $E(\sum_{i=1}^N m_i)$. As mentioned in Section 1, we measure the performance of a given frequency-assignment algorithm by its carried-traffic function $T(r)$ defined in (1.1), which is the expected number of accepted calls per cell per available frequency:

$$T(r) = \frac{1}{nN} E(m_1 + \cdots + m_N). \tag{3.1}$$

The main result of this section is the following.

Theorem 3.1. *Let $(y_1, y_2, \ldots, y_N)$ be any list of N numbers satisfying $0 \leq y_i \leq 1$ for $i = 1, 2, \ldots, N$. Then for any frequency-assignment algorithm,*

$$T(r) \leq \frac{1}{N}\left(r \sum_{i=1}^{N} y_i + \max_j (N_j - Y_j)\right), \tag{3.2}$$

where $(Y_1, Y_2, \ldots, Y_S)$ is the v–m transform of $(y_1, y_2, \ldots, y_N)$.

Proof: Since the average carried traffic cannot exceed the average offered traffic, and since the offered traffic is uniform and has intensity r, then $E(m_i) \leq rn$ for $i = 1, 2, \ldots, N$. The result now follows from (3.1) and Theorem 2.2. ❐

The following result is a simple corollary to Theorem 3.1 but it it allows us to define the important function $T_0(r)$, which is an upper bound on the carried traffic function for any frequency-assignment algorithm for the G-system.

Theorem 3.2. *Suppose $T_0(r)$ is the value of the following linear program:*

$$\frac{1}{N}\left(r \sum_{i=1}^{N} y_i + y_{N+1}\right) = \textit{minimum, subject to} \tag{3.3}$$

$$0 \leq y_i \leq 1 \qquad i = 1, 2, \ldots, N \tag{3.4}$$

$$\sum_{i=1}^{N} y_i a_{ij} + y_{N+1} \geq N_j \qquad j = 1, 2, \ldots, S. \tag{3.5}$$

Then for any frequency-assignment algorithm, $T(r) \leq T_0(r)$.

Proof: If (3.4) is satisfied, then by Theorem 3.1, the bound (3.2) holds. If now y_{N+1} is a real number satisfying (3.5), then by the definition (2.1) of the v–m transform,

$$y_{N+1} \geq N_j - Y_j, \qquad j = 1, 2, \ldots, S, \tag{3.6}$$

Thus from (3.2), it follows that

$$T(r) \leq \frac{1}{N}\left(r\sum_{i=1}^{N} y_i + y_{N+1}\right) \tag{3.7}$$

for any set of numbers $y_1, y_2, \ldots, y_{N+1}$ satisfying (3.4) and (3.5). This completes the proof. ❐

We can illustrate Theorem 3.1 with the graph of Figure 1, for which $(N_1, N_2, N_3) = (3, 3, 1)$. If we take $(y_1, y_2, \ldots, y_7) = (1, 1, 1, 1, 1, 1, 1)$, then $(Y_1, Y_2, Y_3) = (3, 3, 1)$, and $\max_j(N_j - Y_j) = 0$. Thus Theorem 3.1 implies $T(r) \leq r$ for all $r \geq 0$. If $(y_1, y_2, \ldots, y_7) = (0, 0, 0, 0, 0, 0, 0)$, then $(Y_1, Y_2, Y_3) = (0, 0, 0)$, and $\max_j(N_j - Y_j) = 3$. Thus by Theorem 3.1, $T(r) \leq 3/7$ for all $r \geq 0$. Finally if $(y_1, y_2, \ldots, y_7) = (0, 0, 1, 1, 1, 1, 0)$, then $(Y_1, Y_2, Y_3) = (2, 2, 0)$, and $\max_j(N_j - Y_j) = 1$, so that Theorem 3.1 implies $T(r) \leq (4/7)r + (1/7)$ for all $r \geq 0$. Combining these three inequalities, we find that the carried traffic funcion for the graph of Figure 1 must lie in the shaded region of Figure 2. In the next section, we will see that no further improvement on these bounds is possible.

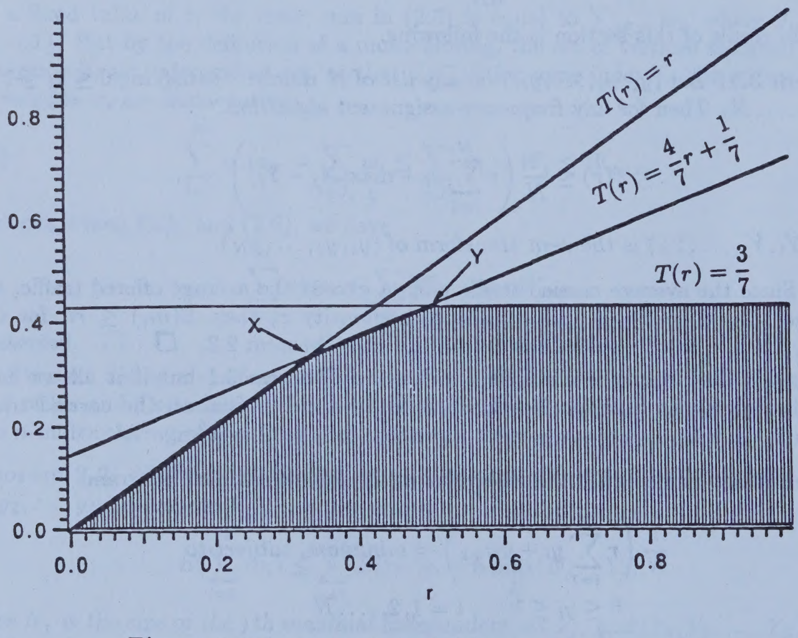


Figure 2. The region bounded by the three inequalities $T(r) \leq r$, $T(r) \leq 3/7$, and $T(r) \leq (4/7)r + (1/7)$. The boundary of this region is the $T_G(r)$ curve for the system of Figure 1.

4. Asymptotic Performance of Fixed Frequency Assignment Algorithms.

In this section we will study the asymptotic performance of a class of frequency-assignment algorithms which we call *fixed frequency-assignment algorithms.* By asymptotic we mean that n, the number of available frequencies, is large. We shall not be precise about the underlying model of the offered traffic except to require that it satisfy the following "asymptotic traffic property (ATP)," which can be defined by the performance of a simple one-cell frequency-assignment algorithm.

Suppose then that there is just one cell, and that there are n available frequencies. An obvious frequency-assignment algorithm in this situation is a "greedy" algorithm, i.e., one in which when a new call request arrives, it is assigned to any unoccupied frequency, if there is one, and otherwise it is blocked. If the offered traffic is k Erlangs, we denote by $C(k,n)$ the carried traffic, i.e., the expected number of occupied frequencies for the greedy algorithm. The ATP referred to above is

$$\lim_{n\to\infty} \frac{C(rn,n)}{n} = \min(r,1) \qquad \text{for all } r \geq 0. \tag{4.1}$$

The ATP says that if the offered traffic is less than the number of available frequencies, then asymptoticaly all call requests can be honored, whereas if the offered traffic exceeds the number of available frequencies, then asymptotically all available frequencies will be occupied. It is thus a kind of law of large numbers. Most common traffic models satisfy the ATP, including the standard Poisson arrivals with exponential call durations. (The "Erlang B formula" can be used to prove this—see [1], Sec. 3.4.3).

We will now define a family of frequency-assignment algorithms, which we call *fixed frequency-assignment algorithms,* and proceed to analyze their asymptotic performance, assuming the ATP.

Thus let $\mathbf{X} = (X_1, X_2, \ldots, X_S)$ be a list of S numbers satisfying $X_j \geq 0$ and $\sum_j X_j = 1$. We use these numbers to define an n-multicoloring of G, as follows. For $j = 1, 2, \ldots, S$, we define $n_j = nX_j$, and partition the n available frequencies into S classes $C_1, C_2, \ldots, C_S$, with class C_j containing exactly n_j frequencies. Then for $j = 1, 2, \ldots, S$, we assign each of the frequencies in C_j to each vertex in the jth maximal independent set V_j. In this way we obtain an n-multicoloring of G with exactly $m_i = \sum_j n_j a_{ij} = n \sum_j X_j a_{ij} = nx_i$ frequencies assigned to vertex v_i, where $(x_1, x_2, \ldots, x_n)$ is the m-v transform of $\mathbf{X}$. The frequency-assignment algorithm associated with this n multicoloring is simply this: when a call request arrives at v_i, assign it one of the m_i frequencies available at v_i, if at least one is not in use; otherwise, block the call. We call this the $\mathbf{X}$ fixed frequency-assignment algorithm. It is, in effect, N independent greedy algorithms, one for each cell in the system.

Theorem 4.1. *Assume the ATP. If $T_{\mathbf{X}}(r)$ denotes the carried traffic function of the* $\mathbf{X}$ *fixed frequency-assignment algorithm, then*

$$\lim_{n\to\infty} T_{\mathbf{X}}(r) = \frac{1}{N}\sum_{i=1}^{N} \min(r, x_i) \qquad \textit{for all } r \geq 0. \tag{4.2}$$

Proof: In the $\mathbf{X}$ fixed frequency-assignment algorithm, each of the N cells operates independently of the others. Within the ith cell, the algorithm is a greedy algorithm with $x_i n$ available frequencies and the offered traffic is rn Erlangs. Thus the carried traffic in the ith cell is $C(rn, x_i n)$, and the carried traffic for the entire system is

$\sum_{i=1}^{N} C(rn, x_i n)$, so that

$$T_{\mathbf{X}}(r) = \frac{1}{nN} \sum_{i=1}^{N} C(rn, x_i n). \tag{4.3}$$

But from the ATP property (4.1),

$$\lim_{n \to \infty} \frac{1}{n} C(rn, x_i n) = \min(r, x_i) \tag{4.4}$$

Combining (4.3) and (4.4) we obtain (4.2). ❐

Theorem 4.2. *Suppose $T_1(r)$ is the value of the following linear program:*

$$\frac{1}{N} \left(\sum_{j=1}^{S} N_j X_j - \sum_{i=1}^{N} z_i \right) = \text{maximum, subject to} \tag{4.5}$$

$$X_j \geq 0 \qquad j = 1, 2, \ldots, S \tag{4.6}$$

$$z_i \geq 0 \qquad i = 1, 2, \ldots, N \tag{4.7}$$

$$\sum_{j=1}^{S} X_j = 1 \tag{4.8}$$

$$\sum_{j=1}^{S} X_j a_{ij} - z_i \leq r \qquad i = 1, 2, \ldots, N \tag{4.9}$$

Then for any fixed r, there exists a fixed frequency-assignment algorithm whose asymptotic carried traffic function is arbitrarily close to $T_1(r)$.

Proof: We begin with vectors $\mathbf{X} = (X_1, X_2, \ldots, X_S)$ and $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ whose components satisfy (4.6), (4.7), (4.8), and (4.9). Let $(x_1, x_2, \ldots, x_N)$ be the m-v transform of $\mathbf{X}$. We note that (4.9) is equivalent to $x_i - z_i \leq r$, and so by (4.7) and (4.9) we have $x_i - z_i \leq \min(r, x_i)$. Therefore

$$\sum_{i=1}^{N} (x_i - z_i) \leq \sum_{i=1}^{N} \min(r, x_i). \tag{4.10}$$

Note also

$$\begin{aligned} \sum_{i=1}^{N} x_i &= \sum_{i=1}^{N} \sum_{j=1}^{S} X_j a_{ij} \\ &= \sum_{j=1}^{S} X_j \sum_{i=1}^{N} a_{ij} \\ &= \sum_{j=1}^{S} X_j N_j, \end{aligned} \tag{4.11}$$

so that

$$\sum_{i=1}^{N}(x_i - z_i) = \sum_{j=1}^{S} X_j N_j - \sum_{i=1}^{N} z_i. \tag{4.12}$$

Therefore we have

$$\begin{aligned}
\lim_{n\to\infty} T_{\mathbf{X}}(r) &= \frac{1}{N}\left(\sum_{i=1}^{N}\min(r, x_i)\right) \quad \text{by Theorem 4.1}\\
&\geq \frac{1}{N}\sum_{i=1}^{N}(x_i - z_i) \quad \text{by (4.10)}\\
&= \frac{1}{N}\left(\sum_{j=1}^{S} N_j X_j - \sum_{i=1}^{N} z_i\right) \quad \text{by (4.12)} \qquad (4.13)
\end{aligned}$$

Thus the **X** fixed frequency-assignment algorithm has a carried traffic function which is asymptotically at least as large as the objective function (4.5), and this proves the theorem. ❐

To illustrate Theorem 4.1, we return to the graph of Figure 1. If we let $\mathbf{X}_1 = (1/3, 1/3, 1/3)$, then its *m-v* transform is $\mathbf{x}_1 = (1/3, 1/3, 1/3, 1/3, 1/3, 1/3, 1/3)$, and Theorem 4.1 implies that the $\mathbf{X}_1$ algorithm's asymptotic carried traffic curve is $T_1(r) = \min(r, 1/3)$. Similarly, if $\mathbf{X}_2 = (1/2, 1/2, 0)$, then $\mathbf{x}_2 = (1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 0)$, and by Theorem 4.1, the $\mathbf{X}_2$ algorithm's asymptotic carried traffic curve is $T_2(r) = (6/7)\min(r, 1/2)$. More generally, by taking all possible convex combinations of $\mathbf{X}_1$ and $\mathbf{X}_2$, i.e., vectors of the form $\mathbf{X} = \lambda\mathbf{X}_1 + (1-\lambda)\mathbf{X}_2$ where $0 \leq \lambda \leq 1$, we obtain a family of curves whose convex hull is the curve bounding the shaded region in Figure 2. Since we saw in Section 3 that no point outside of this shaded region is achievable by any frequency-assignment algorithm, we are justified in asserting that the shaded region in Figure 2 is the achievable region for the performance of frequency-assignment algorithms for the graph of Figure 1. In the next section we will see that this is no accident, but an instance of a general rule.

5. Equality of $T_0(r)$ and $T_1(r)$. General Properties of this Function.

In Section 3 we showed that for any frequency-assignment algorithm for the graph G, the corresponding carried traffic function was bounded above by $T_0(r)$. On the other hand, in Section 4, we showed that if n is sufficiently large, then the performance of certain fixed frequency-assignment algorithms is bounded *below* by $T_1(r)$. Interestingly, however, these two functions are equal.

Theorem 5.1. *$T_0(r) = T_1(r)$, for all $r \geq 0$.*

Proof: By Theorem 3.2, $T_0(r)$ is the value of a certain linear program; and by Theorem 4.2, $T_1(r)$ is the value of another linear program. However, these programs are dual programs (see [2], Section 1.2), and so by the Duality Theorem of Linear Programming ([2], Section 1.8), the values of these two programs are equal, provided both programs are feasible. It is easy to show that both programs are feasible: A feasible solution for the $T_0(r)$ program is $y_1 = y_2 = \cdots = y_N = 0$ and $y_{N+1} = \max_j N_j$; and a feasible solution for the $T_1(r)$ program is $X_1 = 1, X_2 = \cdots = X_S = 0$ and $z_i = a_{i1}$ for $i = 1, 2, \ldots, N$. Thus $T_0(r) = T_1(r)$, as asserted. ❐

Let us denote the common value of the functions $T_0(r)$ and $T_1(r)$ by $T_G(r)$. The next theorem gives the most important general properties of this function.

Theorem 5.2. *The function $T_G(r)$ has the following properties.*

(a) $T_G(r)$ is nondecreasing, continuous, piecewise linear, and convex $\cap$.

(b) $T_G(r) = r$ for all sufficiently small r.

(c) $T_G(r) = \max_j \frac{N_j}{N}$ for all sufficiently large r.

Proof: (a). According to Theorem 4.2, a feasible solution for the $T_1(r)$ program will also be feasible for all $r' \geq r$, and so $T_1(r') \geq T_1(r)$, for all $r' \geq r$. Thus $T_G(r)$ is nondecreasing. Furthermore, the $T_1(r)$ program is a parametric linear program (with parameter r) in the sense of ([2] Section 1.9), and so by a result proved there (page 70), the function $T_1(r)$ is continuous, convex $\cap$, and piecewise linear.

(b). In the $T_0(r)$ program given in Theorem 3.2, let $y_1 = y_2 = \cdots = y_N = 1$, and $y_{N+1} = 0$. This is a feasible solution, and the corresponding value of the objective function is r. Thus by Theorem 3.2,

$$T_G(r) = T_0(r) \leq r \qquad \text{for all } r \geq 0. \tag{5.1}$$

On the other hand, in Theorem 4.1, let $X_1 = X_2 = \cdots = X_S = 1/S$. Then for $i = 1, 2, \ldots, N$, x_i equals $1/S$ times S_i, the number of maximal independent sets containing v_i. Thus $x_i \geq \min_i(S_i/S) > 0$. It follows then from Theorem 4.1 that $\lim_{n\to\infty} T_{\mathbf{X}}(r) = r$ for all $r \leq \min_i(S_i/S)$, and so

$$T_G(r) = T_1(r) \geq r \qquad \text{for all } r \leq \min_i(S_i/S). \tag{5.2}$$

Combining (5.1) and (5.2), we see that $T_G(r) = r$ for all $r \leq \min_i(S_i/S)$, as asserted.

(c). In the $T_0(r)$ program given in Theorem 3.2, let $y_1 = y_2 = \cdots = y_N = 0$ and $y_{N+1} = \max_j N_j$. This is a feasible solution, and the corresponding value of the objective function is $\max_j(N_j/N)$. Thus by Theorem 3.2,

$$T_G(r) = T_0(r) \leq \max_j(N_j/N) \qquad \text{for all } r \geq 0. \tag{5.3}$$

On the other hand, in the $T_1(r)$ program given in Theorem 4.2, let $j = j_0$ be any index for which N_j is largest, and set $X_{j_0} = 1$, $X_j = 0$ for $j \neq j_0$, and $z_i = 0$ for $i = 1, 2, \ldots, N$. This is a feasible solution for all $r \geq 1$, and the corresponding value of the objective function is $\max_j(N_j/N)$, so that

$$T_G(r) = T_1(r) \geq \max_j(N_j/N) \qquad \text{for all } r \geq 1. \tag{5.4}$$

Combining (5.3) and (5.4), we see that $T_G(r) = \max_j(N_j/N)$, for all $r \geq 1$, as asserted.
□

Since the function $T_G(r)$ is the value of the pair of dual linear programs given in Theorems 3.2 and 4.2, it can be computed numerically, using the the simplex method (as described in [2], Section 1.11 or [6], Section 10.8). Although S, the number of maximal independent sets, can grow exponentially with N, the number of vertices, nevertheless we have found the computation of $T_G(r)$ to be feasible for moderately large system sizes, say $N \leq 30$. For example, consider the 21-cell system shown in Figure 3, which is taken from [3].

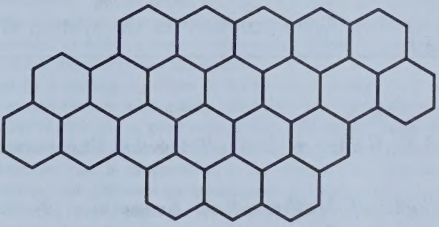


Figure 3. A 21–cell system.

Here the assumption is that two cells whose centers are at a distance of three or less cannot use the same frequency. (The distance between adjacent cell centers is assumed to be 1.) We can show that the function $T_G(r)$ for this system is as shown in Figure 4. To get a feeling for the value of this result, suppose we wished to design a frequency-assignment algorithm for the system in Figure 3 using $n = 96$ frequencies, with a blocking probability of 3%. Then from Equation (1.1) with $P_{\text{block}} = .03$, we find that $T(r) = .97r$, and from Figure 4, we find that for this system, $T_G(r) = .97r$ for $r = .08795$. Thus from Theorem 3.2, the maximum carried traffic under these circumstances is $.97 \times .08795 \times 96 \times 21 = 172.3$ Erlangs. On the other hand, an explicit dynamic frequency-assignment algorithm, called "Remax2," which was introduced in [7] and discussed in [4], achieves a carried traffic of 145 Erlangs for this system. Similarly with $n = 400$ and $P_{\text{block}} = 3.4\%$, Remax2 achieves 696 Erlangs of carried traffic, while the $T_G(r)$ function implies that no more than 719 Erlangs are possible. These and similar computations indicate that the performance bounds developed in this paper, and the actual performance of Remax2, are both quite good.

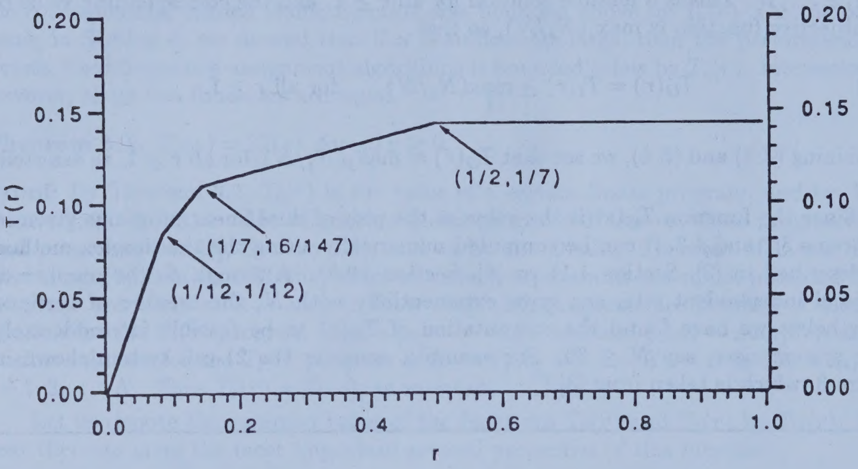


Figure 4. The $T_G(r)$ function for the system of Figure 3.

6. References.

[1] Bertsekas, D. and R. Gallager, *Data Networks.* Englewood Cliffs, N.J. : Prentice-Hall, 1987.

[2] Franklin, J. *Methods of Mathematical Economics.* New York: Springer-Verlag, 1980.

[3] Gamst, A., "Some Lower Bounds for a Class of Frequency Assignment Problems," *IEEE Trans. Veh. Tech.*, vol. VT-35 (Feb. 1986), pp. 8–14.

[4] McEliece, R. J., K. Sivarajan, and J. Ketchum, "Dynamic Frequency Assignment in Cellular Radio," Proc. 1990 IEEE Vehicular Technology Conference, pp. 631–637.

[5] Pennotti, R. J., *Channel Assignment in Cellular Mobile Telecommunication Systems*, Ph. D. Thesis, Polytechnic Institute of New York, 1976.

[6] Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipies in C.* Cambridge: Cambridge U. Press, 1988.

[7] Sivarajan, K., *Spectrum Efficient Frequency Assignment for Cellular Radio.* Ph.D. Thesis, California Institute of Technology, 1990.

NEURAL NETWORK IMPLEMENTATION OF THE SHORTEST PATH ALGORITHM FOR TRAFFIC ROUTING IN COMMUNICATION NETWORKS

Stelios C.A. Thomopoulos[1,#], Lei Zhang[@], Chin Der Wann[#]

[#]Decision and Control Systems Laboratory
Department of Electrical & Computer Engineering
The Pennsylvania State University, University Park, PA 16802

[@]Department of Electrical Engineering
The University of Maryland, College Park, MD 20740

Abstract

A neural network computation algorithm is introduced to solve for the optimal traffic routing in a general N-node communication network. The algorithm chooses multilink paths for node-to-node traffic which minimize a certain cost function (e.g. expected delay). Unlike the algorithm introduced earlier in this area, knowledge of the number of links (hops) between each origin-destination pair is not required by the algorithm, therefore it can be applied to a more variable length path routing problems. The neural network structure for implementing the algorithm is a modification to the one used by the Traveling Salesman algorithm. Computer simulations in a nine- and sixteen-node grid network show that the algorithm performs extremely well in single and multiple paths.

I. Introduction

The computational power and the speed of collective analog networks of neurons in solving optimization problems have been demonstrated by Hopfield and Tank [1]-[3] through the famous "Traveling Salesman Problem". A similar procedure can be applied to solve a number of optimization problems [6]. In order to solve a practical optimization problem using a neural network structure, it is necessary to find algorithms for determining the connections and weights of the neural network so that it converges to the appropriate answer. In this paper, we suggest a neural network structure that can determine the optimal route for node-to-node traffic in an N-node communication network. The structure is an implementation of the so called "Shortest Path Routing Algorithm" in which a route is selected for every origin-destination (OD) pair such that the transmission cost is minimized if data is transmitted along this route.

The main function performed by a routing algorithm is the selection of routes for various origin-destination pairs. There are two main performance measures that are substantially affected by the routing algorithm, the throughput (quantity of service) and the average delay (quality of service). A good routing algorithm should select the routes which have minimum average delay (thus allow more traffic into the network). In the shortest path algorithm, a cost is associated with every link in the network. In most cases, the cost is proportional to the delays. The objective is to find a multilink path joining two nodes that has minimum total cost. Different implementations of the shortest paths algorithm, in both synchronous and asynchronous fashion, are available [4]. In this paper we consider two different NN implementations of the shortest paths algorithm using the actual delay and the derivative delay as cost functions. The neural network structure of the algorithm was first introduced by Rauch and Winarske [5]. Their method, however, has serious limitations. It can find the shortest path for a given OD pair only when the number of links that the path contains is known, which is an unrealistic assumption. A modified structure is suggested in the present paper so that the algorithm can work for arbitrary and unknown number of links in a given OD pair. The NN that is presented in this paper was first introduced in [7].

II. Problem Statement

Consider a N-node network and assume that the connectivity of the network is known. Let c_{ij} denote the capacity of the link connecting node i with node j. If there is no direct connection between node i and j, $c_{ij} = 0$. Therefore, the network can be described by an NxN capacity matrix C with entries c_{ij}. In addition, if every link in the network is a two-way link and has the same capacity in each direction, C is symmetric.

Our problem is to find the path connecting origin and destination nodes that minimizes a cost function such as the expected delay. Since the expected delay across a link is a function of the link capacity c_{ij} and the actual link traffic f_{ij}, several functions can be used to calculate the link cost [4], [5]. For example, the link cost w_{ij} can be determined by

$$w_{ij} = f_0 + [\Sigma f_{ij} / (c_{ij} - \Sigma f_{ij})]^{p} \tag{1.1}$$

where f_0 is the transmission time for each link, and Σf_{ij} is the total flow from all OD-pairs on the link ij. The exponent p can take any positive value, but commonly used values are 1 or 2. The value p = 1 was used in the simulations. The link capacity $c_{ij} - \Sigma f_{ij}$ in (1.1) is the residual capacity in the network when paths for multiple OD-pairs are considered. When the optimal paths for multiple OD-pairs are determined sequentially, the residual link capacity is determined by

$$c_{ij} - \Sigma f_{ij} = c_{ij} - \Sigma f_{ij}(\text{previous OD pairs}) - \Sigma f_{ij}(\text{current OD-pair}) \tag{1.2}$$

With p = 1 in (1.1), two different approaches were taken to solve for the shortest path. In the first approach, which will be referred to as *delay cost* approach, the link cost was computed directly using (1.1). In the second approach, which will be referred to as *derivative delay cost* approach, the link cost was equated to the derivative of w_{ij} in (1.1), i.e.

---

[1] *This work was partially supported by SDIO/IST under Contract N-00014-k-0515 managed by ONR*

$$w_{ij} = f_0 + c_{ij} / (c_{ij} - \Sigma f_{ij})^2 \quad (1.3)$$

which is the link cost that is used in the conventional, optimal solution of the shortest path problem assuming convex and double-differentiable delay (cost) function [4]. The differences in the numerical solutions obtained under the two cost functions (1.1) and (1.3) are discussed in the simulations.

Let the NxN matrix W with entries w_{ij} denote the cost matrix associated with the network. Notice that if there is no direct link between node i and j, $w_{ij} = \infty$ ($c_{ij} = 0$). If this is the case, a very large number is assigned to w_{ij} in the simulation.

To illustrate the problem, consider the 5-node network in Figure 1. The number beside each link represents the corresponding link cost (w_{ij}). The cost matrix W associated with this network is given in Table 1, where L is some large positive number.

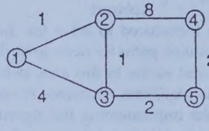


Figure 1 A 5-Node Network

Table 1 Cost Matrix for the 5-Node Network of Fig. 1

	1	2	3	4	5
1	L	1	4	L	L
2	1	L	1	8	L
3	4	1	L	L	2
4	L	8	L	L	2
5	L	L	2	2	L

The shortest path from node 1 to node 5 is obviously 1-2-3-5 and the minimum total cost is $w_{12} + w_{23} + w_{35} = 1+1+2 = 4$.

In the next section we reformulate the shortest path algorithm using a neural network structure.

III. Neural Network Computation Algorithm

In their paper [5], Rauch and Winarske suggested that the solution of the shortest path algorithm can be represented by a 2-dimensional neuron array $V = (V_{ij})$ with each output of the neuron in the array having value $V_{ij} = 0$ or 1. The number of rows in the array is equal to N, the number of nodes in the network, and the number of columns is equal to the number of nodes that the path contains. For the 5-node network of Figure 1, the shortest path connecting node 1 and node 5 can thus be represented by

	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	0
5	0	0	0	1

(2)

It is obvious that for the array to represent a valid path, there can be only one nonzero entry in each column and there can be at most one nonzero entry in each row (this condition is different from the one required by the TSP problem). An nonzero entry in the ijth position of the array can be interpreted as "node i is the jth node in path". Using this representation, a total NxM neurons are needed to represent all the paths having length (number of nodes in the path) M. Given an OD pair, the first and the last column of the array are fixed, so there are Nx(M-2) active neurons in the array which are free to be updated.

As we have mentioned in the previous section, this representation has its limitations. The problem with this representation is that if we do not know how many nodes the shortest path would contain, i.e. M is unknown. Rauch and Winarske assumed that the minimum number of links between a given OD pair could be obtained in advance from the capacity matrix C, in which case M is equal to that number plus one. However, by choosing M this way, we may not be able to find the shortest path because it is possible that a longer path can have lower total cost than that of a shorter path. In our 5-node example, the minimum number of links between node 1 and 5 is 2. If we choose M = 3, the 5x3 array can only give the path that contains 3 nodes, which is 1-3-5 with cost 6. We know though from the previous discussion that the shortest path is 1-2-3-5 with cost 4. It is obvious that the solution given by Rauch-Winarske's (R-W) method is not the correct one.

To overcome the limitations of the R-W method, we fix the number of columns in the array at N, which is the maximum possible number of nodes any path could contain in an N-node network. By doing so, the neuron array (NxN now) can represent all the paths containing N nodes. Since most of the paths have length less than N, we should convert these paths into length N paths and maintain their total cost at the same time in order for them to be represented by the NxN array. This can be achieved ***by adding some zero-cost pseudo links to those "shorter" paths, i.e. paths with less-than-N-links, until their length is equal to N. To implement this idea, for each node we introduce one zero-cost pseudo link that connects the node to itself.*** The traffic can then circle at any node along the path through these pseudo links without increasing the total cost of the path. For the 5-node example of Fig. 1, the network after introducing 5 pseudo links is shown in Fig. 2. Table 2 gives the associated cost matrix.

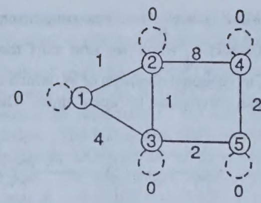


Figure 2 A 5-Node Network with Pseudo Links

Table 2 Cost Matrix for the 5-Node Network of Fig. 2

	1	2	3	4	5
1	0	1	4	L	L
2	1	0	1	8	L
3	4	1	0	L	2
4	L	8	L	0	2
5	L	L	2	2	0

By comparing Table 1 with Table 2, one can see that the only difference between the two cost matrices is that the diagonal elements now become zero instead of a large number L. Using this modified representation, one of the possible solutions to the 5-node network problem with node 1 being the origin and node 5 the destination is

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 1 \end{array} \qquad (3)$$

(3) shows that the shortest path between node 1 and 5 is 1-1-2-3-5, which can be interpreted as 1-2-3-5. Note that the representation of the shortest path is not unique; solutions 1-2-2-3-5, 1-2-3-3-5, and 1-2-3-5-5 all represent the same path.

For a solution to be valid, we require that there is only one nonzero entry in each column and the total number of nonzero entries in the array is equal to N. Under these constraints, the **energy function associated with the network** can be defined as

$$E = (A/2) \sum_k \sum_i \sum_j V_{ik} w_{ij} V_{j,k+1} + (B/2) \sum_k \sum_i \sum_j V_{ik} V_{jk} + (C/2)\Big(\sum_i \sum_j V_{ij} - N \Big)^2 \qquad (4)$$

where the first triple summation gives the total cost from the origin to destination; the second and third terms are the constraints imposed on the output on the neuron array to make it converge to a valid path; A,B, and C are positive enforcement factors.

From Equation (4) we can obtain the connection weight between the ij^{th} neuron and the mn^{th} neuron in the array

$$T_{ij,mn} = - A\, w_{im} (\delta_{n,j+1} + \delta_{n,j-1}) - B\, \delta_{jn} (1-\delta_{im}) - C \qquad (5)$$

where δ_{ij} is the Kronecker's delta, i.e. $\delta_{ij} = 1$ if $i=j$ and $\delta_{ij} = 0$ if $i \neq j$.

The state of the ij^{th} neuron, y_{ij}, can be described by the differential equation

$$dy_{ij}/dt = - y_{ij}/\tau + \sum_m \sum_n T_{ij,mn} V_{mn} + I_{ij} \qquad (6)$$

and

$$V_{ij} = g(y_{ij}) = [\,1 + \tanh(y_{ij}/y_0)\,]\,/2 \qquad (7)$$

$$I_{ij} = Cn \text{ (input bias term)} \qquad (8)$$

for $1 \leq i \leq N, 2 \leq j \leq N-1$.

In high gain limit, the output of the neuron, V_{ij}, is close to 0 or 1, and the energy function defined by (4) will be minimized (it could be a local minimum) when the system reaches its steady state.

IV. Simulation Results

The neural network routing algorithm developed in the previous section was simulated using 9-node and 16-node networks with different link cost assignments. The 9-node grid network shown in Figure 3 was used for the first pilot simulation. All links were assumed to be two-way links and have the same capacity. Under this assumption, the capacity matrix C is symmetric. We also assumed that the initial link cost w_{ij} is inverse proportional to the link capacity c_{ij} because we do not have any knowledge about the link flow (traffic) f_{ij} when we first start the algorithm. So, the cost matrix W is also symmetric and all links have the same cost. The diagonal elements of W, which correspond to pseudo links, are all zero, and a large number is assigned to the elements which correspond to "open" links. The cost matrix used in the pilot simulations is given in Table 3.

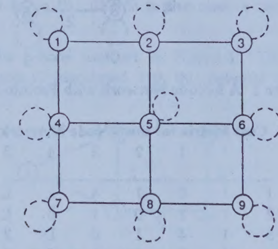


Figure 3 A 9-Node Grid Network with Pseudo Links (dashed lines)

Table 3 Cost Matrix for the 9-Node Grid Network of Fig. 3

	1	2	3	4	5	6	7	8	9
1	0	2	20	2	20	20	20	20	20
2	2	0	2	20	2	20	20	20	20
3	20	2	0	20	20	2	20	20	20
4	2	20	20	0	20	20	2	20	20
5	20	2	20	2	0	2	20	2	20
6	20	20	2	20	2	0	20	20	2
7	20	20	20	2	20	20	0	2	20
8	20	20	20	20	2	20	2	0	2
9	20	20	20	20	20	2	20	2	0

In the pilot simulations only one OD-pair was considered for the given cost matrix. For each given OD pair, the first and the last column in the neuron array are fixed. The states of the rest N(N-2) (= 63 in the 9-node network) active neurons are updated according to the steady state expression of Equations (6) and (7). The initial value of the output of each active neuron is a random number uniformly distributed in [0, 2/N] such that

$$E\{ \sum_i \sum_j V_{ij} \} = N \qquad (9)$$

We start the network with low gain, i.e., the slope of the hyperbolic tangent curve in Equation (7) is small (y_0 large). This choice would allow the system to find better minima of the energy surface. After 100 iterations, we start slowly increasing the gain (decreasing y_0) until the system converges and the values of V_{ij} are near 0 or 1. The results for the 9-node pilot study were obtained with the following parameters.

$A = 20, B = C = 500, n = 9.5, \tau = 1$

$y_0 = 250$ (initial), $y_0 = 20$ (final)

The algorithm is sensitive to these parameters, since a bad operating point may result in divergence (oscillation).

Table 4 shows the shortest path found by the algorithm between node 1 and node 9; (a) is the initial condition, (b) is the result after 100 iterations and (c) the result after 200 iterations (final result). The shortest path found is 1-1-2-2-2-2-5-6-9, which can be interpreted as 1-2-5-6-9. Table 5 gives similar results for a different OD pair.

Because of the symmetry of the grid network, the shortest path is not unique for some OD pairs, which makes the convergence more difficult. The algorithm will find one of the shortest paths depending on the initial conditions. In the simulation, we also noticed that if the gain is fixed at a higher value right from the beginning, the system is very easy to get stuck at some local minima. By starting at low gain and slowly increasing it, we have, so far, been able to reach the global minimum on all single-OD-pair trials for the pilot 9-node grid network.

In practice, the link cost w_{ij} in a communication network depends on the actual traffic going through that link. To obtain the actual traffic distribution for the entire network, the algorithm should be repeated for every OD pairs (there are Nx(N-1) of them). After the actual traffic conditions in the network become available, the cost matrix W can be updated by using equations (1.1) and (1.2). The algorithm is then repeated for each OD pair again, and the optimal path is found for each OD pair that will prevent some links from becoming too crowded. By so repeating the algorithm, the ANN could eventually obtain the optimal flow distribution for the network in the sense that the expected delay on the entire network is minimized for a given set of link capacities. This approach was used to obtain shortest paths for multiple OD pairs in a 9- and 16-node networks.

After the pilot simulation was successfully completed, the ANN algorithm was tested in multiple OD-pairs in both 9-node and 16-node networks. In the 9-node network, the algorithm was tested with four different OD-pairs. Each link was assumed to have normalized capacity 0.5, whereas the flow on each OD-pair was taken to be 0.1. The "optimal" paths were obtained sequentially by presenting to the ANN one OD-pair at a time. The initial conditions on A, B, C, n, and t that were used in the pilot simulation, were also used in these simulations. The annealing temperature schedule was slightly different; the initial temperature was kept the same at y_0 (initial) = 250, but the final temperature was set at y_0(final) = 30.70193 for all trials. After the network converged to an "optimal" path for a given OD-pair, the link cost was updated according to Eq. (1.1) with p = 1. The four chosen OD-pairs were: A = (1,8), B = (2,8), C = (4,2), and D = (7,3) [The first number in the parenthesis indicates the origin, while the second the destination]. For each OD-pair, convergence was achieved after 200 iterations, in agreement with the pilot simulation. The initial and final neural activations for each OD-pair are shown in Table 6. In this particular simulation, the order that the OD-pairs were presented in the ANN was ABCD and a ***single path*** was assumed to carry all the traffic for each OD pair. The same initial conditions (neural activation) were used for each OD-pair. The "optimal" path that was obtained was A = (1-4-5-8), B = (2-5-8), C = (4-1-2), and D = (7-4-5-2-3), with total cost 16.63968. This path is not the overall optimal path which has cost 15.42510 (see Table 7), but is very close to it.

In order to determine the effect of the sequence at which the different OD-pairs are presented to the neural network, all possible permutations in the sequence of the four OD-pairs were presented and the "optimal" paths were recorded. Table 7 summarizes the different "optimal" paths and the frequency they occurred. When the same initial conditions were used for each OD-pair, the set of paths 1, with total cost 16.63968, very close to the optimal set of paths 10 with cost 15.42510, was obtained 66.67% of the times. When the initial conditions (neural activations) for each OD-pair were chosen randomly, the frequency of path set 1 dropped to 29.17%. However, the frequency of the optimal path set 10 increased from 0.0%, that it was when the same initial conditions were used, to 12.50%. The effect of the initial conditions is currently investigated. From the results obtained so far, it appears that the different initial conditions result in a more even distribution of the path sets among low cost path sets than the distribution of the path sets obtained with fixed initial conditions. Table 8 summarizes the correspondence between the sequence with which the four OD-pairs were presented to the network and the path set that the NN converged to under fixed initial conditions and different initial conditions. The numbers of the path sets correspond to the path set numbers of Table 7.

In order to determine the stability of the "optimal" path sets, two OD-pairs were alternatingly presented to the NN and the path sets were recorded. The chosen OD-pairs were A = (1,9) and B = (8,3). The link capacity was kept the same, i.e. 0.5 units per link, but the input data flow was raised to 0.25 data units. Starting with zero initial neural activation, fixed for each OD-pair, the NN converged to a stable solution in one iteration. Furthermore, it converged to the same path set, irrespective of which OD-pair was presented first (columns 1 and 2, Table 9). When the initial neural activation were random but fixed for all OD-pairs, the solution was stabilized in a few iterations, columns 3 and 4 in Table 9. The same path sets were obtained irrespective of what OD-pair was presented first. However, when different random initial activation was used each time a new OD-pair was presented, the path sets stabilized after a few presentations at slightly different set paths, depending on which OD-pair was presented first. In this particular experiment, all the different path set that were obtained are equivalent from cost point of view.

To test the ability of the ANN to optimize the network performance further by creating multi-path routes for multiple OD pairs, a comparative study was conducted by allocating different percentages on the total flow on each path and repeating the algorithm by interleaving the different OD pairs until the total traffic from all OD-pairs was accommodated. The simulations were conducted using both the *delay link cost* [Eq. (1.1)] as well as the *derivative delay link cost* [Eq. (1.3)]. For a single OD pair but different percentage of traffic allocation at each "shortest" path, the results for the two different cost functions applied to a 9-node network are summarized in Table 10 and Fig. 4. From these results, it can be seen that smaller increments per iteration result is lower total cost, in general. Furthermore, the derivative delay cost function (1.3) yields paths that slightly outperform those obtained by the delay cost function (1.1) for most increments. However, the differences are not so significant. One advantage of the derivative delay cost function is that the number of loops observed is the "shortest" paths was eliminated completely in the run cases. A small percentage of "shortest" paths, usually less than 5%, occasionally contained loops when the delay cost function was used instead. The existence of loops is currently under investigation.

An identical simulation to the one described in the previous paragraph was conducted for three OD pairs in a nine node network. The results for the delay and derivative delay cost functions are summarized in Fig.s 5 and 6 respectively. Similar conclusions to the single-path experiment can be drawn; lower increments per iteration result in lower total cost. The derivative delay cost yields slightly better results than the delay cost function itself. Analytical statistics of the number of times each path appeared as different increments of flow were used to obtain the shortest paths are given in Tables 11 and 12 for 100%, 25%, and 1% increments per iteration. The amount of flow each path carries is also shown on the tables. As it is seen, most of the traffic flow is concentrated in a few "good" paths as the size of flow increment decreases. Furthermore, the derivative delay cost yields a slightly lower final cost (delay) than the delay cost.

The NN routing algorithm was also tested in a 16-node square grid network with link capacity 0.5 units, the same as in the 9-node network. Four OD-pairs were used to test the NN. The test OD-pairs were: A = (1,8), B = (2,12), C = (14,4), D = (1,13). When the OD-pairs were presented to the NN in the ABCD sequence, the optimal path set was obtained after 200 iterations for each OD-pair, Table 13. The same annealing schedule as in the 9-node case was used. Note that the path set in Table 13 is globally optimal. Due to space limitations, initial conditions, intermediate results after 100 iterations, and final results after 200 iterations are only given for OD-pair 4. For the other three OD-pairs only cumulative, final results are given. The sensitivity of the solution to the order at which the different OD-pairs are presented in the NN is being investigated. Furthermore, the appearance of paths that contain closed loops which seem to appear when the cost of looping is low and the path of the initially presented OD pair splits the network graph into two separate subgraphs, is also being investigated. Additional details on the simulation results on a 16-node network can be found in [8].

V. Conclusions

In this paper, a neural-based computational algorithm has been developed for solving optimal traffic routing problems in communication networks. **The key idea in this algorithm is the introduction of pseudo links which allow the extension of any path to a length N path so that it can be represented by an NxN neuron array.** The proposed NN algorithm can be used to obtain optimal routes for multiple OD-pairs by presenting to the NN one OD-pair at a time. The sequential presentation of OD-pairs in the NN guarantees a unique path for each OD-pair. Once a "shortest" path is obtained it can be used to carry the entire traffic for a given OD pair, provided its capacity is not exceeded. A more even distribution of the flow from different OD pairs using multi-paths is obtained by allocating only a percentage of the total flow from a given OD pair to a "shortest" path, and repeating the algorithm by interleaving the different OD pairs. An implementation of the algorithm using incremental, circular presentations that can allow multiple paths per OD-pair has been tested numerically and found to reducing the routing cost (i.e. total delay). Computer simulation results on a 9-and 16- node grid networks show that the algorithm performs well when the slope of the nonlinearity curve (characteristics of the neuron) is slowly increased during iterations. The "optimal" path sets were found to be close to the global optimal and be stable independent of the sequence the OD-pairs were presented to the NN. Simulation results in a 16-node network indicate that the NN algorithm continues to perform well in larger networks. The performance of the algorithm in upscaled, multi-node networks with different connectivity is presently being evaluated.

References

[1] J.J. Hopfield and D.W. Tank, *"Neural" Computation of Decisions in Optimization Problems*, Biological Cybernetics, Vol. 54, pp. 141-152, 1985.

[2] J.J. Hopfield, *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, Proc. Natl. Acad. Sci. USA, Vol. 79, pp. 2554-58, April 1982.

[3] J.J. Hopfield, *Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons*, Proc. Natl. Acad. Sci. USA, Vol. 81, pp. 3088-3092, May 1984.

[4] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, 1987.

[5] H.E. Rauch and T. Winarske, *Neural Networks for Routing Communication Traffic*, IEEE Control Systems Magazine, pp. 26-30, April 1988.

[6] E.W. Page and G.A. Tagliarini, *Algorithm Development for Neural Networks*, SPIE Symposium on Innovative Science and Technology, Los Angeles, CA, Jan. 1988.

[7] L. Zhang, and S. C. A. Thomopoulos, *Neural Network Implementation of the Shortest Path Algorithm for Traffic Routing in Communication Networks*, Joint International Conference on Neural Networks, JICNN '89, June 1989, Washington, D.C., Poster Paper Proceedings.

[8] S. C. A. Thomopoulos, and C. D. Wann, "*Neural Network Implementation of the Shortest Path Algorithm for Traffic Routing in Communication Networks*, Technical Report, April 1990, Decision and Control Systems Laboratory, Dept. of ECE, The Pennsylvania State Univ., University Park, PA 16802.

Table 4 Simulation Results for the 9-Node Grid Network of Fig. 3 (Origin = Node 1; Destination = Node 9)

Node	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	1.000	0.060	0.222	0.056	0.011	0.114	0.038	0.208	0.000
2	0.000	0.127	0.090	0.211	0.079	0.138	0.082	0.134	0.000
3	0.000	0.010	0.059	0.215	0.001	0.077	0.196	0.191	0.000
4	0.000	0.046	0.197	0.049	0.089	0.020	0.047	0.113	0.000
5	0.000	0.164	0.011	0.171	0.037	0.048	0.115	0.015	0.000
6	0.000	0.028	0.071	0.085	0.216	0.060	0.048	0.159	0.000
7	0.000	0.144	0.127	0.176	0.113	0.100	0.054	0.135	0.000
8	0.000	0.188	0.047	0.019	0.104	0.031	0.081	0.207	0.000
9	0.000	0.005	0.046	0.158	0.080	0.029	0.015	0.209	1.000

(a) Initial Conditions

Node	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	1.000	0.661	0.169	0.085	0.057	0.008	0.008	0.001	0.000
2	0.000	0.289	0.752	0.790	0.767	0.733	0.016	0.008	0.000
3	0.000	0.003	0.011	0.044	0.049	0.009	0.058	0.001	0.000
4	0.000	0.012	0.004	0.001	0.001	0.009	0.001	0.006	0.000
5	0.000	0.003	0.012	0.054	0.095	0.149	0.756	0.009	0.000
6	0.000	0.000	0.001	0.001	0.001	0.011	0.012	0.634	0.000
7	0.000	0.000	0.000	0.000	0.000	0.001	0.002	0.001	0.000
8	0.000	0.000	0.000	0.001	0.001	0.008	0.003	0.203	0.000
9	0.000	0.000	0.000	0.000	0.000	0.001	0.011	0.019	1.000

(b) After 100 Iterations

Node	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	1.000	0.994	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.000	0.000	0.994	0.999	0.999	0.995	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	0.940	0.000	0.000
6	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.937	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000

(c) After 200 Iterations

Shortest Path: 1-1-2-2-2-2-5-6-9 (1-2-5-6-9)

Table 5 Simulation Results for the 9-Node Grid Network of Fig.3 (Origin = Node 3; Destination = Node 8)

Node	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	0.000	0.060	0.222	0.056	0.011	0.114	0.038	0.208	0.000
2	0.000	0.127	0.090	0.211	0.079	0.138	0.082	0.134	0.000
3	1.000	0.010	0.059	0.215	0.001	0.077	0.196	0.191	0.000
4	0.000	0.046	0.197	0.049	0.089	0.020	0.047	0.113	0.000
5	0.000	0.164	0.011	0.171	0.037	0.048	0.115	0.015	0.000
6	0.000	0.028	0.071	0.085	0.216	0.060	0.048	0.159	0.000
7	0.000	0.144	0.127	0.176	0.113	0.100	0.054	0.135	0.000
8	0.000	0.188	0.047	0.019	0.104	0.031	0.081	0.207	1.000
9	0.000	0.005	0.046	0.158	0.080	0.029	0.015	0.209	0.000

(a) Initial Conditions

Node	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	0.000	0.004	0.035	0.049	0.036	0.008	0.004	0.001	0.000
2	0.000	0.650	0.771	0.780	0.761	0.625	0.103	0.006	0.000
3	1.000	0.264	0.108	0.065	0.039	0.008	0.004	0.001	0.000
4	0.000	0.000	0.001	0.001	0.001	0.008	0.011	0.004	0.000
5	0.000	0.004	0.039	0.067	0.109	0.253	0.757	0.703	0.000
6	0.000	0.011	0.001	0.001	0.001	0.008	0.011	0.004	0.000
7	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.009	0.000
8	0.000	0.000	0.001	0.001	0.001	0.008	0.020	0.187	1.000
9	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.009	0.000

(b) After 100 Iterations

Node	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.000	0.979	0.998	0.998	0.998	0.978	0.000	0.000	0.000
3	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	0.977	0.975	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

(c) After 200 Iterations

Shortest Path: 3-2-2-2-2-2-5-5-8 (3-2-5-8)

Table 6 Simulation results from a 9-Node Grid Network of Fig. 3
with four OD-pairs: (1,8), (2,8), (4,2), (7,3)

Cumulative Results

OD-pair in order of presentation	*Initial Temperature*	*Temperature after 100 iterations*	*Final Energy after 200 iterations*	*Selected Path at the end of annealing*
(1,8)	250	30.70193	12.94737	**1-4-5-8**
(2,8)	250	30.70193	13.87045	**2-5-8**
(4,2)	250	30.70193	14.79352	**4-1-2**
(7,3)	250	30.70193	**16.63968**	**7-4-5-3**

The total energy is: 16.63968

Initial and final (200iterations) neural activation for the third OD-pair (4,2)

Initial neural activation

0.000	0.139	0.014	0.014	0.196	0.027	0.078	0.056	0.000
0.000	0.206	0.110	0.088	0.011	0.061	0.176	0.047	1.000
0.000	0.099	0.219	0.216	0.077	0.047	0.057	0.141	0.000
1.000	0.167	0.068	0.062	0.170	0.179	0.038	0.149	0.000
0.000	0.029	0.046	0.142	0.116	0.165	0.055	0.129	0.000
0.000	0.000	0.130	0.150	0.108	0.010	0.117	0.033	0.000
0.000	0.154	0.125	0.011	0.222	0.179	0.049	0.063	0.000
0.000	0.184	0.145	0.149	0.102	0.021	0.180	0.191	0.000
0.000	0.158	0.171	0.097	0.164	0.212	0.160	0.013	0.000

Neural activation after 200 iterations

0.000	0.000	0.000	0.000	0.000	0.903	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.983	0.995	1.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.000	0.995	0.995	0.995	0.976	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Results	Selected OD pairs				Path cost	% (with same initial condition)	% (with different initial conditions)
	A=(1,8)	B=(2,8)	C=(4,2)	D=(7,3)			
1	1-4-5-8	2-5-8	4-1-2	7-4-5-2-3	16.63968	66.67%	29.17%
2	1-4-5-8	2-5-8	4-5-2	7-4-5-2-3	19.01330	12.50%	---
3	1-4-5-8	2-5-8	4-5-2	7-4-5-6-3	17.68074	8.33%	---
4	1-2-5-8	2-1-4-5-8	4-5-2	7-4-5-2-3	19.93636	8.33%	---
5	1-2-5-8	2-1-4-5-8	4-5-2	7-4-5-6-3	18.60382	4.17%	---
6	1-4-5-8	2-5-8	4-1-2	7-4-5-6-3	16.34818	---	4.17%
7	1-4-5-8	2-5-8	4-5-2	7-4-1-2-3	16.63968	---	12.50%
8	1-2-5-8	2-5-8	4-1-2	7-4-5-2-3	17.68074	---	16.67%
9	1-2-5-8	2-5-8	4-1-2	7-4-5-6-3	16.34818	---	16.67%
10	1-2-5-8	2-5-8	4-5-2	7-4-5-6-3	15.42510	---	12.50%
11	1-2 5-8	2-5-8	4-5-2	7-4-1-2-3	17.68074	---	4.17%
12	1-4-7-8	2-5-8	4-5-2	7-4-5-6-3	16.34818	---	4.17%

Table 7 Summary of the simulation results on four OD pairs

Sequence of OD pairs	Results		Sequence of OD pairs	Results	
	with same initial condition	with different initial conditions		with same initial condition	with different initial conditions
A B C D	1	1	C A B D	3	10
A B D C	1	1	C A D B	2	10
A C B D	3	7	C B A D	5	11
A C D B	2	7	C B D A	4	7
A D B C	1	1	C D A B	2	10
A D C B	1	1	C D B A	4	12
B A C D	1	9	D A B C	1	8
B A D C	1	9	D A C B	1	8
B C A D	1	9	D B A C	1	8
B C D A	1	1	D B C A	1	1
B D A C	1	9	D C A B	1	8
B D C A	1	6	D C B A	1	1

Table 8 Shortest paths obtained for the different sequences of four OD pairs

Sequence of OD pairs		All zero initial condition for All OD pairs		Same initial condition for all OD pairs		Different initial condition for each OD pair	
1.	1 -- 9	111255569		111444589		111222589	
2.	8 -- 3	852222333	852222223	855555523	855555563	855555563	855555563
3.	1 -- 9	111222569	111222569	111444569	111444589	111444589	111114589
4.	8 -- 3	852222333	852222333	888555523	855555523	855555563	874444523
5.	1 -- 9	111222569	111222569	111444569	111444589	111255569	144444589
6.	8 -- 3	852222333	852222333	888555523	855555523	855552223	855555563
7.	1 -- 9	111222569	111222569	111444569	111444589	111222569	111225589
8.	8 -- 3	852222333	852222333	888555523	855555523	855555563	855555563
9.	1 -- 9	111222569	111222569	111444569	111444589	111114589	111444589
10.	8 -- 3	852222333	852222333	888555523	855555523	888555523	855555563

Table 9 Shortest paths from alternatingly presentation of two OD pairs in a 9-node network using different initial conditions. First column under each initial conditions corresponds to initialization with a different OD pair.

Percentage	Final Cost for Delay Cost Function Case	Final Cost for Derivative Delay Cost Function Case
1.0%	3.053078175	3.055994987
2.0%	3.046160793	3.042394257
4.0%	3.050959682	3.062650584
5.0%	3.071259595	3.066443539
10.0%	3.058737946	3.049192429
12.5%	3.092867184	3.107277489
16.7%	3.041032504	3.036917877
20.0%	3.070995522	3.111513518
25.0%	3.114019965	3.087734604
33.3%	3.138028240	3.095095252
50.0%	3.199145318	3.191892244
100.0%	3.507691004	3.420241559

Table 10 Final cost from a 9-Node Network with different flow increments

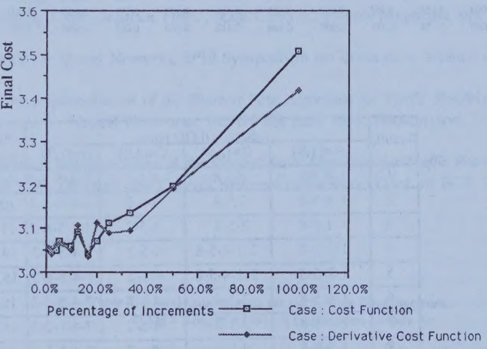


Figure 4 Final Cost vs. Percentage of Flow Increment

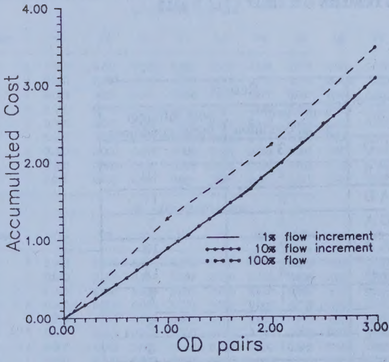


Figure 5 Accumulated Cost vs. OD pairs (Delay cost function case)

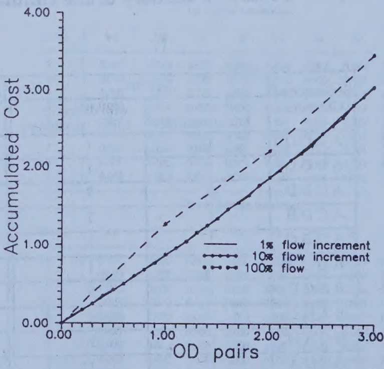


Figure 6 Accumulated Cost vs. OD pairs (Derivative Delay cost function case)

Table 11 Number of times a given path appeared in 1/(flow increment) trials using different flow increments. ***Case 1: Delay Cost Function***

```
--------------------------------------------
100% Flow Increment per Iteration
              PAIR: 1
times
path
appeared Flow     <  Path   >
  1  0.1200000  9 8 5 4 1 0 0 0 0
--------------------------------------------
              PAIR: 2
       Flow     <  Path   >
  1  0.1200000  2 1 4 7 0 0 0 0 0
--------------------------------------------
              PAIR: 3
       Flow     <  Path   >
  1  0.1200000  8 5 2 3 0 0 0 0 0
The final cost is:  3.44939137
--------------------------------------------
25% Flow Increment per Iteration
              PAIR: 1
       Flow     <  Path   >
  1  0.0300000  9 8 5 4 1 0 0 0 0
  2  0.0600000  9 6 5 4 1 0 0 0 0
  1  0.0300000  9 8 5 2 1 0 0 0 0
--------------------------------------------
              PAIR: 2
       Flow     <  Path   >
  1  0.0300000  2 5 4 7 0 0 0 0 0
  3  0.0900000  2 1 4 7 0 0 0 0 0
--------------------------------------------
              PAIR: 3
       Flow     <  Path   >
  1  0.0300000  8 7 4 5 6 3 0 0 0
  2  0.0600000  8 5 2 3 0 0 0 0 0
  1  0.0300000  8 5 6 3 0 0 0 0 0
 The final cost is:  3.10797119
--------------------------------------------
1% Flow Increment per Iteration
              PAIR: 1
       Flow     <  Path   >
  9  0.0108000  9 8 5 2 1 0 0 0 0
  12  0.0144000  9 8 5 4 1 0 0 0 0
  16  0.0192000  9 6 5 2 1 0 0 0 0
  55  0.0659998  9 6 5 4 1 0 0 0 0
  1  0.0012000  9 8 5 4 1 4 1 0 0
  5  0.0060000  9 6 5 4 1 4 1 0 0
  1  0.0012000  9 8 7 4 1 0 0 0 0
  1  0.0012000  9 6 9 6 5 4 1 0 0
--------------------------------------------
              PAIR: 2
       Flow     <  Path   >
  28  0.0335999  2 1 4 7 0 0 0 0 0
  63  0.0755995  2 5 4 7 0 0 0 0 0
  9  0.0108000  2 5 8 7 0 0 0 0 0
--------------------------------------------
              PAIR: 3
       Flow     <  Path   >
  46  0.0551999  8 5 6 3 0 0 0 0 0
  49  0.0587999  8 5 2 3 0 0 0 0 0
  2  0.0024000  8 7 4 5 2 3 0 0 0
  2  0.0024000  8 7 4 5 6 3 0 0 0
  1  0.0012000  8 7 4 1 2 3 0 0 0
 The final cost is:  3.05268764
--------------------------------------------
```

Table 12 Number of times a given path appeared in 1/(flow increment) trials using different flow increments. ***Case 1: Derivative Delay Cost Function***

```
--------------------------------------------
100% Flow Increment per Iteration
              PAIR: 1
--------------------------------------------
times
path
appeared   Flow     <  Path   >
  1   0.1200000  9 8 5 4 1 0 0 0 0
--------------------------------------------
              PAIR: 2
        Flow     <  Path   >
  1   0.1200000  2 1 4 7 0 0 0 0 0
--------------------------------------------
              PAIR: 3
        Flow     <  Path   >
  1   0.1200000  8 5 6 3 0 0 0 0 0
 The final cost is:  3.44939137
--------------------------------------------
25% Flow Increment per Iteration
              PAIR: 1
        Flow     <  Path   >
  1   0.0300000  9 8 5 4 1 0 0 0 0
  1   0.0300000  9 6 5 2 1 0 0 0 0
  2   0.0600000  9 6 5 4 1 0 0 0 0
--------------------------------------------
              PAIR: 2
        Flow     <  Path   >
  1   0.0300000  2 1 4 7 0 0 0 0 0
  3   0.0900000  2 5 4 7 0 0 0 0 0
--------------------------------------------
              PAIR: 3
        Flow     <  Path   >
  2   0.0600000  8 5 6 3 0 0 0 0 0
  2   0.0600000  8 5 2 3 0 0 0 0 0
The final cost is:  3.07753086
--------------------------------------------
1% Flow Increment per Iteration
              PAIR: 1
        Flow     <  Path   >
  2   0.0024000  9 8 5 2 1 0 0 0 0
  69  0.0827993  9 6 5 4 1 0 0 0 0
  10  0.0120000  9 8 5 4 1 0 0 0 0
  17  0.0204000  9 6 5 2 1 0 0 0 0
  2   0.0024000  9 8 7 4 1 0 0 0 0
--------------------------------------------
              PAIR: 2
        Flow     <  Path   >
  52  0.0623999  2 5 4 7 0 0 0 0 0
  7   0.0084000  2 5 8 7 0 0 0 0 0
  41  0.0491999  2 1 4 7 0 0 0 0 0
--------------------------------------------
              PAIR: 3
        Flow     <  Path   >
  37  0.0443999  8 5 6 3 0 0 0 0 0
  56  0.0671998  8 5 2 3 0 0 0 0 0
  5   0.0060000  8 7 4 5 2 3 0 0 0
  2   0.0024000  8 7 4 1 2 3 0 0 0
 The final cost is:  3.03548908
--------------------------------------------
```

Table 13 Simulation results from a 16-Node Grid Network with four OD-pairs: (1,8), (2,12), (14,4), (1,13)

Cumulative Results

OD-pair in order of presentation	*Initial Temperature*	*Temperature after 100 iterations*	*Final Energy after 200 iterations*	*Selected Path at the end of annealing*
(1,8)	250	30.70193	25.26316	**1-2-3-7-8**
(2,12)	250	30.70193	25.52632	**2-6-7-11-12**
(14,4)	250	30.70193	28.68826	**14-10-6-7-3-4**
(1,13)	250	30.70193	29.63563	**1-5-9-13**

The total energy is: 29.63563

Intermediate and final results for the fourth OD-pair (1,13)

```
-----------------------------------
initial   states   for   OD   pair                  4
-----------------------------------
1.000   0.040   0.111   0.097   0.096   0.092   0.083   0.079   0.049   0.091   0.086   0.002   0.019   0.092   0.092   0.00
0.000   0.004   0.096   0.115   0.041   0.047   0.116   0.057   0.049   0.096   0.100   0.061   0.101   0.118   0.043   0.00
0.000   0.112   0.056   0.111   0.005   0.018   0.034   0.100   0.043   0.004   0.049   0.008   0.105   0.114   0.047   0.00
0.000   0.019   0.118   0.097   0.010   0.029   0.093   0.041   0.101   0.120   0.023   0.108   0.078   0.085   0.082   0.00
0.000   0.072   0.002   0.001   0.111   0.087   0.055   0.071   0.072   0.065   0.017   0.042   0.007   0.020   0.067   0.00
0.000   0.105   0.020   0.093   0.007   0.024   0.115   0.036   0.118   0.084   0.104   0.056   0.092   0.050   0.093   0.00
0.000   0.119   0.040   0.032   0.088   0.043   0.084   0.014   0.034   0.054   0.102   0.007   0.012   0.057   0.069   0.00
0.000   0.010   0.107   0.120   0.004   0.069   0.098   0.108   0.015   0.040   0.043   0.022   0.032   0.043   0.117   0.00
0.000   0.051   0.025   0.025   0.067   0.070   0.103   0.093   0.089   0.038   0.115   0.123   0.055   0.090   0.083   0.00
0.000   0.097   0.073   0.086   0.017   0.108   0.017   0.092   0.013   0.079   0.040   0.016   0.002   0.039   0.049   0.00
0.000   0.039   0.097   0.073   0.080   0.008   0.106   0.122   0.113   0.124   0.095   0.034   0.078   0.022   0.124   0.00
0.000   0.121   0.036   0.025   0.112   0.061   0.091   0.087   0.007   0.076   0.086   0.038   0.037   0.048   0.068   0.00
0.000   0.080   0.029   0.070   0.019   0.071   0.005   0.012   0.027   0.002   0.109   0.059   0.076   0.006   0.008   1.00
0.000   0.102   0.070   0.118   0.018   0.001   0.078   0.059   0.023   0.042   0.109   0.087   0.002   0.043   0.093   0.00
0.000   0.119   0.053   0.121   0.052   0.091   0.038   0.007   0.081   0.097   0.049   0.117   0.074   0.073   0.124   0.00
0.000   0.065   0.105   0.002   0.119   0.028   0.100   0.098   0.086   0.082   0.021   0.015   0.004   0.101   0.004   0.00
-----------------------------------
iteration:                     100   temp:      250.0000         pair:                  4
-----------------------------------
1.000   0.205   0.119   0.074   0.071   0.070   0.072   0.070   0.069   0.065   0.064   0.054   0.044   0.006   0.004   0.00
0.000   0.011   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.000   0.00
0.000   0.722   0.739   0.776   0.779   0.782   0.776   0.769   0.762   0.761   0.758   0.766   0.730   0.682   0.009   0.00
0.000   0.004   0.052   0.061   0.068   0.066   0.065   0.065   0.066   0.065   0.066   0.057   0.048   0.006   0.004   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.001   0.000   0.000   0.001   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.000   0.00
0.000   0.004   0.050   0.059   0.065   0.064   0.064   0.064   0.066   0.068   0.074   0.082   0.134   0.218   0.824   0.00
0.000   0.000   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.001   0.012   0.001   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.001   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.001   0.001   0.001   0.001   0.011   0.026   1.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.011   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.001   0.000   0.00
-----------------------------------
iteration:                     200   temp:      30.70193         pair:                  4
-----------------------------------
1.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.985   0.996   0.996   0.995   0.995   0.995   0.995   0.995   0.995   0.995   0.995   0.994   0.981   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.928   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   1.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.00
The   energy   of   the   network   after   previous   OD   pairs:          29.63563
```

The final energy is: 29.63563

Ergodicity of a Queueing Network and an Alternative Proof of the Maxflow-Mincut Theorem

by

Leandros Tassiulas, Anthony Ephremides
Department of Electrical Engineering and Systems Research Center
University of Maryland

Abstract

We consider a queueing network with state dependent routing and service control. The region of the arrival and service rates, inside which the system stability is quarranted, is derived. An optimal routing policy, which uses local state information, is specified. Next the maxflow problem in deterministic networks is considered. An independent proof of the maxflow-mincut theorem is given by constructing for each deterministic flow network an appropriate queueing network and applying to it the stability results we obtained earlier.

1. Introduction

Consider a queueing network consisting of M queues and L classes of arriving customers. It is represented by a directed graph G (fig. 1) which contains one node (black) for each class of arriving customers, one node (white) for each queue and one destination node (D). The customers of class l arrive according to a Poisson process with rate λ_l. Upon arrival they join one of the queues of the set S_l which contains all the terminal queues of the arcs originating from the node that correspond to class c. Each queue i posesses a single exponential server, which may idle or provide service with rate μ_i. A customer of queue i after its service completion is routed to one of the queues of the set S_i that contains all the terminal nodes of the arcs originating at queue i. If node D is contained in the set S_i then the served customers of queue i may be routed out of the system. We assume that the routing of customers upon arrival in the network is controled and the routing decision may depend on the lengths of all queues of the system,that is the system state, at the arrival time instant t. Similarly the routing of a customer completing service at queue i is controlled and the routing decision is based on the system state at the time instant of service completion. Finally we assume that the service rate of queue i, which can take two values 0 and μ_i, is controlled and the decision rule may be a function of the lengths of all queues of the system.

The objective is to route the customers and control the service rate such that we achieve stability of the system in the sense that the queue length distribution converges, as $t \to \infty$. We wish to find control policies that stabilize the system for a wide range of arrival and service rates. A control policy π is considered "better" than another policy π' if the system is stable under π whenever is stable under π'. We find an optimal policy in this sense and we give a necessary and sufficient condition for stability.

The main motivation for considering this problem is the design of simple distributed routing policies that work well with the channel access protocols arising in multihop radio networks. By "work well" we mean maximize the end to end throughput. Dill and Silvester in [2] propose some multiaccess protocols in CDMA radio networks where the attempt probabilities for the transmission of a packet residing at node i are functions of the lengths of the queues in the neighboring nodes of node i.

The performance of the protocol is numerically evaluated and is shown to be superior of existing fixed routing protocols. In [6] we consider a *constrained* queueing system that models multihop radio networks and we propose a centralized link activation policy that achieves maximum throughput. The queueing model we study here is different since the routing decisions are taken after service and there are no constraints in simultaneous link activations. More importantly the stability condition that we obtain here is expressed directly in terms of the arrival and service rates.

Another potential application of these results is in the distributed control of manufacturing systems. More specifically, in the case where the parts under processing have to choose a machine among several available at each stage of the processing stage, a policy similar to the one we propose here may stabilize the system in the sense considered by Perkins and Kumar in [5].

After the specification of the model in section 2 we state and prove our main stability results in section 3. In section 4 we consider an application of the stability results obtained in section 3, in deterministic network flow optimization theory. More specifically, we obtain an independent proof of the maxflow-mincut theorem that characterizes the solution of the maxflow problem in deterministic flow networks [4].

2. The model

Let us denote by $x_i(t)$ the number of customers in queue i at time t; the customer receiving service at that time is included. The vector of queue lengths of all queues of the system at time t is $x(t) = (x_i(t) : \ i = 1, ..., M)$ and it takes values in the state space $X = Z_+^M$. The routing of the arriving customers of class l is specified by a function $R_l^c : X \to S_c$ in the sense that an arriving customer of class l at time t joins the queue $R_l^c(x(t-))$. The function R_l^c is called the *routing rule* of class l in the following. A served customer of queue i is routed to one of the queues of the set S_i (or out of the system if $D \in S_i$) according to a function $R_i : X \to S_i$ in the sense that the customer of queue i completing service at time t joins queue $R_i(x(t-))$. The function R_i is the *routing rule* of queue i. Finally the service rate of the server of queue i is controlled according to a function $F_i : X \to \{0, \mu_i\}$; the rate of server i at time t is $F_i(x(t-))$. The function F_i is called the *service control rule* of queue i. Note that the control of the service rate available to the queues provide a kind of flow control. An admissible control policy for the network consists of a collection of routing rules, one for each customer class and each queue, and service control rules, one for each queue of the system. Let G be the collection of all admissible control policies. Note that customers of different classes are treated identically by the control policy; also the service times at of customers of different classes are identically distributed in each queue; Hence customers of different classes are indistinguishable after they enter the network and they differ only in the set of queues that they can join upon arrival.

Remark: If the servers never idle, that is $F_i(x)$ is identically μ_i for all i, and the routing is done by random splitting with some fixed probabilities, not dependent on x, in each queue and each arrival stream, then the resulting network is Jacksonian ([3]).

When the network is operated by a control policy from G, and since the arrivals are Poisson and the service times exponentially distributed, the queue length process x is a Markov chain. The rate $q_{xx'}$ of a transition from a state x to a state x' is as follows

$$q_{xx'} = \begin{cases} \mu_i, & \text{if } F_i(x) = \mu_i,\ x'_i = x_i - 1,\ x'_{R_i(x)} = x_{R_i(x)} + 1 \text{ when } R_i(x) \neq D, \\ & \text{and } x'_j = x_j \text{ for } j \neq i,\ j \neq R_i(x) \\ \lambda_l, & \text{if } x'_{R^c_l(x)} = x_{R^c_l(x)} + 1, \text{ and } x'_j = x_j \text{ for } j \neq R^c_l(x). \end{cases}$$

We define the system to be *stable* if the distribution of the queue length vector $x(t)$ converges as $t \to \infty$ to a probability distribution on X. Since x is a Markov chain for any policy in G the stability of the system is equivalent to ergodicity of x. As we said earlier, the performance of a policy π is characterized be the region in which the operational parameters (arrival and service rates) should lie in order for the system to be stable. Let the *arrival* and *service* rate vectors be $\lambda = (\lambda_l : l = 1, .., L)$ and $\mu = (\mu_i : i = 1, ..., M)$ respectively. The *stability region* C_π of policy π, is the collection of all pairs of vectors (λ, μ) for which the system is stable under policy π. A policy π_1 is defined to be "better" than a policy π_2 if $C_{\pi_1} \supseteq C_{\pi_2}$. Note that every two policies are not necessarily comparable. An optimal policy should have a stability region that is a superset of the stability region of any other policy. Hence an optimal policy it may not necessarily exists. The stability region of the system is defined by $C = \cup_{\pi \in G} C_\pi$. If a pair (λ, μ) belongs to C then it is called *stabilizable.*

Consider the *maximum throughput policy* (MTP) with control rules as follows

$$R_i(x) = \begin{cases} D, & \text{if } D \in S_i \\ \arg\min_{j \in S_i}(x_j), & \text{otherwise,} \end{cases}$$

$$R^c_l(x) = \arg\min_{j \in S_i}(x_j),$$

$$F_i(x) = \begin{cases} 0, & \text{if } x_i \leq \min_{j \in S_i}\{x_j\} \\ \mu_i, & \text{otherwise.} \end{cases}$$

Its name will be justified later. If the minimum above is achieved for more than one queue then the argmin can be any of these queues. Arbitrarily we select in this case the argmin to be equal to the smallest index of the queues that achieve the minimum. Note that in the MTP policy the routing and the service rate control rules for a queue i use for decision making queue length information from the queues of the set S_i only. The queues of this set usually correspond to neighboring nodes of the physical system under consideration. Henceforth a distributed implementation of the policy is readily available. The MTP policy has the optimality property that we mentioned earlier. In the next section we will show the optimality of the MTP policy after we characterize the stability region C of the system.

3. Characterization of C optimality of the MTP policy

Given a set of queues S the set C_S is defined to contain every arrival stream c the customers of which can not be routed outside of S; that is $S_c \subseteq S$ for every $l \in C_S$. The set Q_S is defined to contain every queue i, the served customers of which may be routed outside of S; that is $S_i \not\subseteq S$ for every $i \in Q_S$.

Proposition 3.1:

i) A necessary and sufficient condition for a pair of vectors (λ, μ) to belong to C is that for every set of queues S we have

$$\sum_{l \in C_S} \lambda_l < \sum_{i \in Q_S} \mu_i \qquad (3.1)$$

ii) The policy MTP is optimal in the sense that $C_{MTP} = C$.

The proof of the proposition will be given after the following two lemmas. In the rest of this section, whenever we refer to the process x, we assume that the policy MTP controls the system. We need to characterize first some structural properties of the Markov chain x. We review briefly some standard notions for the classification of the states of a Markov chains. These have been developed mainly in the context of discrete time Markov chains but they are easily generalizable for continuous time chains [1]. A state x_0 is *reachable* by a state x_k if there is a sequence of states x_i, $i = 0, ..., k$ such that $q_{x_i x_{i+1}} > 0$ for $i = 1, ..., k-1$. Two states *communicate* if each one is reachable from the other. The relationship "communicate" is an equivalence relationship. We need to characterize the classes of equivalent states of the chain with respect to this relationship. Note first that x is not necessarily irreducible, that is X is not an equivalence class. The following lemma provides a classification of the states of the process x.

Lemma 3.1: When the MTP policy acts on the queueing system and (3.1) holds for every set of queues S, then the subset of the state space

$$R = \{x : \ x \in X \ and \ x \ can \ be \ reached \ from \ state \ \mathbf{0}\}$$

is the unique closed class of equivalent states for the Markov chain; furthermore any state of the set $X - R$ is transient.

Proof: We show first that the state **0** can be reached from any other state $x \in X$. For each queue i let w_i be the number of hops (links) of the minimum hop path from the node of queue i to node D in the topology graph of the network. Consider the linear function $W(x) = \sum_{i=1}^{M} w_i x_i$ on X. We claim that if for some state $x \in X$ we have $W(x) > 0$ then there exists a transition with positive rate from x to some state x' which is such that $W(x) - W(x') = 1$. Consider the index $d = \arg\min_{\{j:\ x_j > 0\}} \{w_j\}$ which is well defined since $W(x) > 0$. We claim that a service completion at queue d will lead the system in the state x' with the above property. We distinguish the following cases.

Case 1 : $w_d = 1$. Since $w_d = 1$, the queue d may direct customers out of the system; according to the policy MTP all served customers of queue i will be directed out of the system. Hence a service completion at queue d will lead the system in a state x' such that $x'_d = x_d - 1$ and $x'_i = x_i$ for $i \neq d$. Apparently $W(x') = W(x) - 1$.

Case 2 : $w_d > 1$. By the definition of w_i we have that

$$w_i = \begin{cases} 1, & \text{if } D \in \ S_i, \\ 1 + \min_{j \in S_i}\{w_j\}, & \text{otherwise.} \end{cases}$$

Hence there exists a queue l in S_d such that $x_l = 0$ and $w_l = w_d - 1$. A served customer of queue d will join an empty queue in S_d and we can always define appropriatelly the indexing of the queues such that the served customer will join the queue l. Apparently the new state x' will be such that $W(x') \leq W(x) - 1$.

We can easily see now that, by selecting the transitions appropriately, from state x after $W(x)$ transitions, we can eventually reach state x' such that $W(x') = 0$. Apparently $x' = \mathbf{0}$ since for any other state the function W is strictly positive.

By definition of the set R, and from the above result, any of its states communicates with zero; hence any two states of the set communicate as well. Apparently the set R is closed since if a state can be reached by a state in R it can be reached by $\mathbf{0}$ as well and it belongs to R itself. Hence the set R is a closed equivalence class of states. No state outside of R can belong to a closed class of states since any state can reach the state $\mathbf{0} \in R$. Because of that it can not belong to a closed equivalence class of states; the latter condition is necessary for recurence. Hence any state $x \in (X - R)$ is transient.◇

To study the ergodicity of x under the MTP policy we need the following generalization of Foster's ergodicity criterion for nonirreducible Markov chains.

Foster's criterion: Let x(t) be a markov chain with state space X, transition rate matrix Q, and only one closed equivalence class of states with all the rest states being apparently transient. Let q_x be defined by $q_x = \sum_{y \in X} q_{xy} \quad , \quad x \in X$. Then a sufficient condition of ergodicity of x(t) is that we have $\sup_{x \in X} q_x < \infty$ and there exists a function $V : X \to R$, an $\epsilon > 0$ and a finite set $X_0 \subset X$ such that

$$-\epsilon > \sum_{y \in X} \frac{q_{xy}}{q_x} V(y) - V(x) \quad , \quad x \notin X_0$$

and

$$\sum_{y \in X} \frac{q_{xy}}{q_x} V(y) < \infty \quad , \quad x \in X.$$

Lemma 3.2: The stability region C_{MTP} of the system under the MTP policy contains all pairs (λ, μ) such that for every set of queues S (3.1) holds.

Proof: Let q_x be defined by $q_x = \sum_{y \in X} q_{xy}, \quad x \in X$. The transition rate q_{xy} is greater than zero only if the transition from x to y corresponds to an arrival or service completion; hence we have

$$q_x \leq \sum_{c \in C} \lambda_c + \sum_{i \in S} \mu_i \qquad \forall x \in X. \tag{3.5}$$

Consider the function V defined on the state space X of the chain by $V(x) = \sum_{i=1}^{M} (x_i)^2$. We can easily verify that

$$\sum_{y \in X} \frac{q_{xy}}{q_x} V(y) \leq 2V(x) < \infty, \quad x \in X. \tag{3.6}$$

Apparently the set V_b defined by $V_b = \{x : x \in X, \quad V(x) \leq b\}$ has finite cardinality for all b. In the following we will show that for some fixed $\epsilon > 0$ there exists some b, which may be a function of the arrival and service rates, such that

$$-\epsilon \geq \sum_{y \in X} \frac{q_{xy}}{q_x} V(y) - V(x), \quad if \ x \notin V_b. \tag{3.7}$$

Then, from (3.5-3.7) and using Foster's criterion we can conclude that that x is ergodic. By simple calculations in (3.7) we get

$$\sum_{y \in X} \frac{q_{xy}}{q_x} V(y) - V(x) = \sum_{y \in X} \frac{q_{xy}}{q_x} \sum_{i=1}^{M} 2x_i(y_i - x_i) + \sum_{y \in X} \frac{q_{xy}}{q_x} \sum_{i=1}^{M} (y_i - x_i)^2 \tag{3.8}$$

When q_{xy} is different than 0, the transition from x to y corresponds either to an arrival or to a service completion; hence the the states x and y differ in at most two elements and each difference is at most one. In this case we have $\sum_{i=1}^{M}(x_i - y_i)^2 \leq 2$ and the second term in the right hand side of (3.8) is bounded by

$$\sum_{y \in X} \frac{q_{xy}}{q_x} \sum_{i=1}^{M} (x_i - y_i)^2 \leq 2. \tag{3.9}$$

Now we will bound the first term of the sum in the right hand side of (3.8). Let $\lambda_i(x)$ be the sum of the arrival rates over all customer classes which route the customers upon arrival to queue i when the system is in state x. Let $\mu_{ij}(x)$ be defined by

$$\mu_{ij}(x) = \begin{cases} \mu_i, & \text{if } R_i(x) = j \text{ and } F_i(x) = \mu_i \\ 0, & \text{otherwise.} \end{cases}$$

By grouping together, in the first part of (3.8), the terms that correspond to the same queue we have in the following

$$\sum_{y \in X} \frac{q_{xy}}{q_x} \sum_{i=1}^{M} 2x_i(y_i - x_i) = \frac{2}{q_x} \sum_{i=1}^{M} x_i(\lambda_i(x) - \mu_i + \sum_{j=1}^{M} \mu_{ji}(x)) \tag{3.10}$$

When we are in state x consider a permutation $i_1, i_2, ..., i_M$ of the queues which is such that $x_{i_{m-1}} \leq x_{i_m}$, $m = 2, ..., M$. Apparently the permutation is a function of the state but we do not carry this over for notational simplicity. Note that if queue i_l routes the served customers to queue i_m then no queue i_k for $k < m$ belongs to S_{i_l}. In view of this observation the right part of (3.10) can be written as

$$\sum_{i=1}^{M} x_i(\lambda_i(x) - \mu_i + \sum_{j=1}^{M} \mu_{ji}(x)) = \sum_{j=1}^{M} x_{ij}(\lambda_{ij} - \mu_{ij} + \sum_{l=j+1}^{M} \mu_{i_l i_j}). \tag{3.11}$$

By a proper rearrangment of the terms and some calculations we have

$$\sum_{j=1}^{M} x_{i_j}(\lambda_{i_j} - \mu_{i_j} + \sum_{l=j+1}^{M} \mu_{i_l i_j}) = \sum_{j=2}^{M} (x_{i_j} - x_{i_{j-1}})(\sum_{l=j}^{M} \lambda_{i_l} - \sum_{l=j}^{M} \mu_{i_l} + \sum_{m=j}^{M} \sum_{l=m+1}^{M} \mu_{i_l i_m}) +$$

$$x_{i_1}(\sum_{l=1}^{M} \lambda_{i_1} - \sum_{l=1}^{M} \mu_{i_1} + \sum_{m=1}^{M} \sum_{l=m+1}^{M} \mu_{i_l i_m}). \tag{3.12}$$

Consider the sets T_j, $j = 1, ..., L$ defined as

$$T_j = \{i_l : M \geq l \geq j\}.$$

Observe that from the definition of the policy we have

$$\sum_{l=j}^{M} \lambda_{i_l}(x) = \sum_{c \in C_{T_j}} \lambda_c \tag{3.13}$$

and

$$\sum_{m=1}^{M}\sum_{l=m+1}^{M}\mu_{i_l i_m}(x)-\sum_{l=j}^{M}\mu_{i_l}=-\sum_{l\in Q_{T_j}}\mu_l. \tag{3.14}$$

Relations (3.10-3.14) imply that

$$\sum_{y\in X}\frac{q_{xy}}{q_x}\sum_{i=1}^{M}2x_i(y_i-x_i)=$$

$$=\frac{2}{q_x}\sum_{j=2}^{M}(x_{i_j}-x_{i_{j-1}})(\sum_{c\in C_{T_j}}\lambda_c-\sum_{l\in Q_{T_j}}\mu_l)+\frac{2}{q_x}(\sum_{c\in C_{T_j}}\lambda_c-\sum_{l\in Q_{T_j}}\mu_l) \tag{3.15}$$

Consider the number m defined as

$$m=\max_{S\subset\{1,\ldots,M\}}\{\sum_{c\in C_S}\lambda_c-\sum_{i\in Q_S}\mu_i\}. \tag{3.16}$$

From (3.15-3.16) we get

$$\sum_{y\in X}\frac{q_{xy}}{q_x}\sum_{i=1}^{M}2x_i(y_i-x_i)\leq\frac{2}{q_x}mx_{i_M}. \tag{3.17}$$

We can easily see that the relation $V(x)\geq b$ implies that

$$x_{i_M}\geq\sqrt{\frac{b}{M}} \tag{3.18}$$

and for any state x we have

$$q_x\leq\sum_{l=1}^{L}\lambda_l+\sum_{i=1}^{M}\mu_i \tag{3.19}$$

We denote by d the right hand side of (3.19) in the following. From (3.17-3.19) we get

$$\sum_{y\in X}\frac{q_{xy}}{q_x}\sum_{i=1}^{M}2x_i(y_i-x_i)\leq\frac{2}{d}m\sqrt{\frac{b}{M}} \tag{3.20}$$

Equations (3.7),(3.8) and (3.20) give that

$$\sum_{y\in X}\frac{q_{xy}}{q_x}\leq 2+\frac{2}{d}m\sqrt{\frac{b}{M}} \tag{3.21}$$

If condition (3.1) holds, then m will be negative as defined. If in (3.21) we replace b by $M(\frac{d}{2m}(2+\epsilon))^2$ then we get the desired relationship (3.7). ◇

Proof of proposition 3.1: We show first the necessity of part i. Assume that for the set of queues S we have

$$\sum_{l\in C_S}\lambda_l\geq\sum_{i\in Q_S}\mu_i. \tag{3.22}$$

and there exists a policy π_0 that stabilizes the network. Then the Markov chain of queue lengths $x(t)$ under this policy is ergodic. Assume that we start the network with the stationary distribution; then $x(t)$ is a stationary ergodic process and has a stationary distribution. Let $x_S(t)$ be the total number of packets at the queues of the set S at time t. Let $D(t)$, $A(t)$ be the counting processes of departures by any queue of Q_S and arrivals at any queue of the set S respectively, until time t. Then we have

$$x_S(t) = x_S(0) + D(t) - A(t). \tag{3.23}$$

The point process that corresponds to $D(t)$ is stationary and ergodic; its rate varies between 0 and $\sum_{i \in Q_S} \mu_i$ when all the queues of the set Q_S are either empty, or full and route customers out of the system, respectively. Hence the ratio $\frac{D(t)}{t}$ converges, almost surely as $t \to \infty$, to a number $d < \sum_{i \in Q_S} \mu_i$. Similarly the arrival point process is stationary and ergodic. Its rate is always greater than or equal to $\sum_{c \in C_S} \lambda_c$. Hence the ratio $\frac{A(t)}{t}$ converges almost surely to a number $\lambda \geq \sum_{c \in C_S} \lambda_c$. Then from (3.22) and (3.23) we can conclude that $x_S(t)$ goes to infinity as $t \to \infty$ which contradicts the fact that $x(t)$ is ergodic. Lemma (3.3) implies the sufficiency of condition (3.1) in part i. Part ii follows immediatelly from lemma 3.3 and the above necessity result. ◇

4.An alternative proof of the maxflow mincut theorem

A flow network consists of a connectivity graph $G = (V, E)$, a capacity assignment on the links $C : E \to R^+$, a prespecified origin node v_0 and a prespecified destination node v_d. Without loss of generality we assume that there is no edge terminating at node v_0 or originating at node v_d. A *feasible flow* is a vector $\mathbf{f} = (f_e : e \in E)$ that satisfies the capacity constraints $0 \leq f_e \leq C(e)$ and the flow conservation equations

$$\sum_{\substack{e \text{ originates} \\ \text{at } v}} f_e = \sum_{\substack{e \text{ terminates} \\ \text{at } v}} f_e \qquad v \in (V - \{v_0, v_d\}). \tag{4.1}$$

Let F be the set of feasible flows. The *flow transfer* f of a feasible flow $\mathbf{f}$ from node v_0 to v_d is $f = \sum_{\substack{e \text{ originates} \\ \text{at } v_0}} f_e$. The maximum flow problem asks for the maximization of f over the set of feasible flows. That is

$$\max_{\mathbf{f} \in F} f.$$

A flow that achieves the maximum flow transfer is a maxflow. The basic theorem that characterizes the solution of the maximum flow problem is the maxflow-mincut theorem. We need the notion of a *cut* of a flow network in order to state that theorem. This is defined as a as partition of the set of nodes V into two sets W, W' such that the set W contains the node v_0 and the set W' contains the node v_d. A *forward* link of the cut is directed from a node of W to a node of W'. The capacity $C(W, W')$ of the cut equals to the sum of the capacities of the forward links. A mincut is a cut with minimum capacity over all the cuts.

Maxflow-mincut Theorem: In a flow network the flow transfer of a maxflow equals to the capacity of a mincut.

The proof that has been given to this theorem ([4]) is based on duality theory and algorithmic arguments. Here we give an alternative proof which is based on the stability properties of the queueing network that we studied in section 3.

For a given flow network, let say N, we construct a corresponding queueing network Q_N as follows. We consider one queue $q_{(v,w)}$ for each link (v, w) of the flow network. The served customers of queue $q_{(v,w)}$ can be routed to any queue that corresponds to links originated at node w; if w is the destination node v_d then the served customers of $q_{(v,w)}$ may leave the system. There is only one stream of arriving customers with rate λ; the arriving customers can be routed to any queue $q_{(v_0,w)}$ that corresponds to the link (v_0, w) which originates from node v_0. In the following we are going to use interchangebly the links of N and the corresponding queues of Q_N.

Lemma 4.1: If the queueing network Q_N is stabilizable when the arrival rate is λ then there exists a feasible flow $\mathbf{f}$ in the flow network with flow transfer λ.

Proof: If the queueing network is stabilizable then under the MTP policy the Markov chain $x(t)$ has a stationary distribution. We start the network with the stationary distribution and the process $x(t)$ is stationary and ergodic. Consider a vector $\mathbf{f} \in R_+^{|E|}$ such that the element f_e that corresponds to link e equals to the intensity (rate) of the departure process of the queue that corresponds to link e. Note that since x(t) is a stationary process, the departure process of each queue is stationary and the intensity is well defined. We claim that the flow vector $\mathbf{f}$ is a feasible flow for the network N with flow transfer equal to λ. The rate of the departure process in queue i is less than or equal to its service rate μ_i which by definition equals to the capacity of the corresponding link of N. Hence $\mathbf{f}$ satisfies the capacity constraints. Consider all the queues corresponding to links originating at v_0. Any exogeneous arrival is routed to one of these queues. Furthermore these queues receive only exogeneous arrivals. Hence the sum of the arrival rates for the queues originating at v_0 is equal to λ and to the sum of the departure rates from these queues. Consider all links originating from v_0. The sum of their flows is equal to the sum of the departure rates of the corresponding queues, which is equal to the sum of their arrival rates. The later sum equals to the rate λ of the arrival stream and the flow transfer of $\mathbf{f}$ is indeed equal to λ.

It remains to show that $\mathbf{f}$ satisfies the flow conservation equations (4.1). Consider a node $v \in (V - \{v_0, v_d\})$. The sum of the flows of the links originating at v is equal to the sum of the departure rates of the corresponding queues which is equal to the sum of the arrival rates at the same queues. By construction of Q_N, these queues receive traffic only from those queues that correspond to incomming links at node v. Hence the flow conservation equations are satisfied. ◇

Lemma 4.2: The queueing network Q_N is stabilizable if the arrival rate λ is less than the capacity of a mincut of the flow network N.

Proof: We will show that if λ is less than the capacity of a mincut then for every set S of queues the condition (C.1) holds. Then stabilizability follows from proposition 3.2. For every set of queues S consider the set of nodes V_S of N such that the queue that corresponds to any link originating from a node $v \in V_S$ is contained in S and $v_d \notin V_S$. If node v_0 does not belong to V_S that means there exists a link originating at v_0 such that the corresponding queue does not belong to S. Hence C_S is empty and condition (3.1) holds. If v_0 belong to V_S then $\sum_{c \in C_S} \lambda_c = \lambda$. But there exists a cut of N such that for every forward link of the cut the corresponding queue belongs to Q_S. Hence whenever λ is less than the capacity of a mincut (3.1) is satisfied and the lemma follows ◇

Proposition 4.1: The total flow of a maxflow of the network N equals to the capacity of a mincut of the same network.

Proof: It is easy to show that for any flow fand any cut the total flow is less than the capacity of the cut which readily implies that the solution of the maxflow problem should be less than or equal to the capacity of a mincut. By lemmas (4.1), (4.2) we have that for any λ less than the capacity of a mincut there exists a feasible flow with flow transfer λ. Hence the flow transfer of a maxflow should be greater than or equal to any number smaller than the capacity of a mincut. The proposition follows then. ◇

References

[1] J.W.Cohen, "The Single Server Queue", North Holland, 1982.

[2] J.C. Dill, J.A. Silvester, "Adaptive Algorithms for Improving the Throughput of CDMA Packet Radio Networks," Proceedings IEEE Infocom 1989.

[3] J.R.Jackson, "Networks of Waiting Lines", Oper. Res. 5, 518-521.

[4] C. H. Papadimitriou and K. Steiglitc, "Combinatorial Optimization Algorithms and Complexity", Prentice-Hall, Englewood Cliffs, N.J. 1982.

[5] J. Perkins, P.R. Kumar," Stable, Distributed, Real-Time Scheduling of Flexible Manufacturing Assembly/Disassembly Systems," IEEE Trans. on Automat. Contr., Vol.34, pp.139-148, Feb.1989.

[6] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput of Multihop Radio Networks," Proceedings CDC 1990.

[7] J.Walrand, "Queueing Networks", Prentice Hall, 1988.

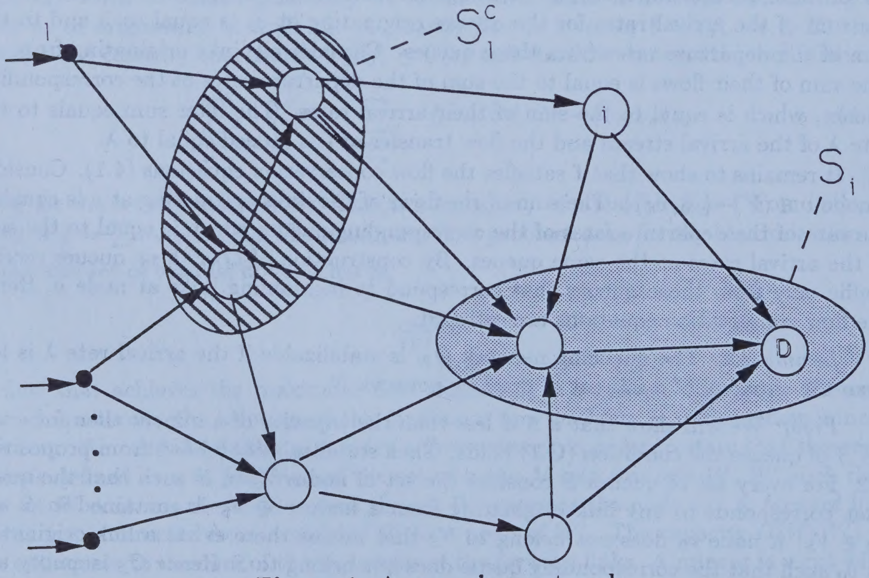


Figure 1 A queueing network.

AN EFFICIENT DISTRIBUTED ALGORITHM FOR ROUTING ON K-DISJOINT PATHS OF MINIMAL TOTAL LENGTH

Chunsiang Cheng
IBM, Austin, TX

Srikanta P.R. Kumar
Northwestern University, Evanston, IL

J.J. Garcia-Luna
SRI International, Menlo Park, CA

Abstract

A distributed asynchronous algorithm is presented for finding K-disjoint paths of minimal total length from a source to a destination, for any given $K > 1$. The paths are found in K iterations of a distributed shortest path algorithm, applying a graph transformation in each iteration. The performance of the algorithm is discussed.

1. INTRODUCTION

In this paper, we present a distributed asynchronous algorithm for finding K node-or-link-disjoint paths of minimum total length from a source to a destination for any $K > 1$. The design of this distributed algorithm is based on the centralized algorithm of Suurballe [9]. The latter involves K iterations, and within each iteration a shortest path from source to sink is found on the graph transformed from the graph of the previous iteration. A principal difficulty in implementing this centralized algorithm in a distributed model is that the graph transformations result in links of zero weights. This precludes the use of the distributed shortest-path algorithms that have been suggested earlier ([2],[5],[6]), as they require all link weights to be strictly positive. We provide an extension of the distributed shortest-path algorithm (based on our prior work [1]) to handle arbitrary link weights and employ this distributed algorithm for the design of the distributed algorithm for finding K-shortest disjoint paths in a network with dynamic topology. (See [7] for a distributed algorithm for $K = 2$).

The paper is organized as follows. Section 2 summarizes previous work, which is the basis of our new algorithm. Section 3 describes the network model. Section 4 introduces a new shortest-path routing algorithm applicable to networks with arbitrary link weights. Section 5 presents the K-DPA algorithm. Concluding remarks are given in Section 6.

2. PREVIOUS WORK

Suurballe [9] proposed an algorithm that reduces the problem of finding the shortest K-disjoint paths from a source to a sink to the problem of finding the shortest path from source to sink K consecutive times. Each iteration of finding the shortest path works on the graph obtained from the previous iteration through a special transformation of link costs. This algorithm may be described inductively as follows.

Let nodes s and t denote the source and the sink, respectively, in a given directed weighted graph $G = (V, E)$. If a link (u, v) is not in E, then it is assumed that its weight is

infinite (∞). Let G^n denote the directed weighted graph for the n^{th} iteration, $1 \leq n \leq K$, with G^1 as the initial graph G. The graphs in different iterations have the same set of nodes and links as in the initial graph; however, the link weights change according to the rules described below. A link from node u to node v in G^n is denoted as $(u,v)^n$, with its weight $w(u,v)^n$. Let p^n be the shortest path from s to t in G^n. Let ρ^n be a set of n disjoint paths from source s to sink t in G such that their total length (sum of the weights of all links in ρ^n) is the minimum among all sets of n disjoint paths from the source s to sink t. Let d_j^n be the shortest distance from node i to destination t in G^n. The transformation is:
(a) $w(u,v)^n = w(u,n)^{n-1} + d_v^{n-1} - d_u^{n-1}$, if (u,v) is not in p^{n-1};
(b) $w(u,v)^n = \infty$, and $w(v,u)^n = 0$, if (u,v) is in p^{n-1}.
The set of shortest disjoint paths ρ^n can be obtained by the operation $p^n \oplus \rho^{n-1}$, where $\oplus$ is an exclusive-or operator that takes the union of p^n and ρ^{n-1} excluding any link (u,v) in p^n if (v,u) is in ρ^{n-1}. The total distance of the K shortest disjoint paths between source s and sink t is $\sum_{n=1}^{k} \sum_{m=1}^{n} d_s^m$. An important property is that each transformed link has nonnegative weight. This ensures that no negative cycles are produced, but zero length cycles can occur. The algorithm terminates at the x^{th} iteration, where x is the connectivity from s to t.

In Section 4, a distributed shortest-path algorithm to handle this is presented when no cycles with negative length exist. We , then, combine graph transformation and the distributed shortest-path algorithm, and avoid the difficulties mentioned above.

3. NETWORK MODEL AND NOTATIONS

The environment for the algorithm is an asynchronous point-to-point network represented by a weighted graph $G(V,L)$, where V is the set of nodes numbered $1, 2, \ldots, N$, and $E \subseteq V \times V$ is the set of links. Each link is bidirectional. Each node is a computing unit involving a processor, a local memory, and also an input queue and an output queue with unlimited capacity. A link from node i to node j is denoted by (i,j); the cost is denoted by d_{ij}. Each node knows only its local environment, that is, its neighbors and the weights of the adjacent links. All input events (messages from neighbors and changes in the cost or status of links) and output events (messages to neighbors) are processed one at a time on a first-come, first-served basis. We assume that messages can be sent and received over a link only when it is functioning. However, a message sent over a link need not arrive at the receiver, as the link may fail during transmission. When the link is functioning, messages can be independently transmitted in either direction, and they arrive at the other end node within a finite time, in sequence and with no errors. Whenever a link fails or recovers or has weight change, each end node is notified in a finite time, but not necessarily at the same time. When a link recovers, there are no messages sent for transmission on the link are deleted after the link fails. A node failure or recovery is taken to be the failure or recovery of all adjacent links [1][2][5][6].

A path from node i to node j via neighbor k of node i, denoted R_{ij}^k, is either a sequence of links, or a sequence of nodes. The length of the path is the sum of the weights of all the edges in that path. A *simple path* from node i to node j is a sequence of nodes with no node repeated more than once. The *head* of a path R_{ij} is defined to be the last node preceding the destination j in the sequence of nodes in R_{ij}. That is, if $R_{ij} = (i, n_1, n_2, \ldots, n_r, j)$, then the head of R_{ij} is n_r if $r > 0$, and equal to i if $r = 0$.

Typically, in distributed routing algorithms, distances are maintained at every to all destinations. Each distance is usually derived by accumulating the weights of a set of links sequentially. This sequence of links will be referred to as the *implicit path* of this distance. Let $D_i = [D_{ij}^k]$ denote the *distance table*, whose rows correspond to destinations and columns to neighbors, stored in node i, with entry D_{ij}^k representing the distance from node i to node j via neighbor k of node i. The *routing table* is an array derived from D_i with one entry for each destination. Each entry of the routing table of node i is a triplet specifying the destination, say j, the preferred neighbor P_{ij} as the neighbor along the shortest path R_{ij} to the destination, and the current shortest distance, minimum of row j in D_i, denoted by $RDIST_i(j)$. There is only one type of message in our distributed shortest-path algorithm called *routing vector* or *update message.* The routing vector (or update message) is a vector of entries, each of which contains a destination node j and a distance, to that destination from the node sending the vector, which is denoted by D_{ij}.

4. A DISTRIBUTED SHORTEST-PATH ALGORITHM FOR ARBITRARY LINK WEIGHTS

The bouncing effect and the counting-to-infinity problem are well-known drawbacks of the widely used BF algorithm in a dynamic network. Any distributed shortest-path algorithm, designed for a network with arbitrary link weights must eliminate these two drawbacks. The distributed shortest-path algorithm in [1] overcomes these drawbacks. (Similar results are suggested in [4] and [8].) However, these require a network with strictly positive link weights. In this section, we describe a new distributed algorithm, derived from [1], which finds the shortest path problem in a network with arbitrary link weights. Here, the shortest paths are obtained from the set of simple paths only. The correctness of this distributed algorithm is defined to mean that the algorithm terminates in finite time after the occurrence of the last topological change, and the distances of the hereditary shortest paths coverage correctly.

Definition 1 *We say that* $R_{ij} = (i, n_1, n_2, \ldots, n_r, j)$*, the shortest path from node* i *to node* j*, is a hereditary shortest path, if* $R_{n_k j} = (n_k, n_{k+1}, \ldots, n_r, j)$ *is also the shortest path from node* n_k *to node* j*, where* $1 \leq k \leq r$*.*

Some shortest paths will be non-hereditary, if negative length cycles exist in network.

A main feature of the algorithm in [1] is the use of the notion of *head of path,* which permits each node to infer the path implicit in a distance and routing table of a node is associated with a head of the path corresponding to this distance entry. Suppose that the design of the algorithm is such that, at all times, the distance and routing table at each node satisfy the following property as a form of local shortest-path consistency: the implicit path of a distance entry D_{ij}^k with associated head $h_{ij}^k = h$ equals the implicit path of the distance entry D_{ih}^k augmented by node j, i.e., from node h to node j. If each column of the distance table and the routing table satisfies this property, then the path implicit in each distance entry can be inferred from the head of the path information above, which in term can be used to maintain simple paths only. The advantage of this approach is that the overhead, incurred over the BF algorithm is only one node identity per table entry.

The routing-table update procedure is such that, at all times, the routing table at any node i satisfies the following conditions:

- c.1 any destination j will be assigned a distance copied from column k iff D_{ij}^k is the minimum among all distances in row j of distance table;

c.2 each node v extracted from D^k_{ij} must be such that D^k_{iv} is also the minimum among all the distances in row v.

c.3 any path extracted path for a distance $RDIST_i(j)$ is the extracted path of distance D^k_{ij} in column k assuming that $RDIST_i(j)$ is copied from D^k_{ij}.

The algorithm termination follows due to the fact that if a distance, whose extracted path has n hops, is sent an infinite number of times, then a distance, whose extracted path has $n-1$ hops, must also be sent an infinite number of times. Recursively applying this argument leads a distance maintained, whose extracted path has zero hop, which is a contradiction. This will be referred to as the CKG algorithm.

5.THE DISTRIBUTED K-DISJOINT PATHS ALGORITHM

Throughout the following description, s denotes the source node, and t denotes the sink node.

Each iteration of Suurballe's algorithm can be divided into two parts: finding a shortest path from node s to node t, and then transforming the graph. To implement this in the distributed model, we can utilize the distributed shortest-path routing algorithm of the previous section. Any distributed algorithm, that implements the iterations of Suurballe's algorithm sequentially, requires the detection of termination of the distributed shortest-path algorithm which consumes extra overhead. In the K-DPA described below, the iterations overlap in time. However, each node is able to distinguish the messages for different iterations via the indications contained in the messages that it received and sent. Each node maintains multiple pairs of distance and routing tables. Specifically, each node keeps n distance and routing tables if it participates in the n^{th} iteration of the computation, where the n^{th} iteration is defined as the computation corresponding to generating and maintaining the distance and routing tables for finding the shortest path from s to t in G^n. In the worst case, a node may have to keep $K+1$ distance and routing tables. For each node, say i, participating in any iteration n, $1 \leq n \leq K$, similar to the distributed shortest-path routing algorithm presented in the previous section, we take the routing table of this node, denoted as RT^n_i, to be a vector, each of whose entries is a quadruple specifying the destination j, the preferred neighbor $P_{ij}(n)$, the current shortest distance $RDIST_{ij}(n)$, and the head $HEAD_{ij}(n)$ of the shortest path from i to j in graph G^n. This quadruple is denoted as $R_{ij}(n)$. The distance table, denoted by DT^n_i, is a matrix, each of whose entries is a pair $(D^k_{ij}(n), h^k_{ij}(n))$ where $D^k_{ij}(n)$ represents the distance from node i to node j via node k, and $h^k_{ij}(n)$ represents the head node of the corresponding path. Infinite distances are denoted by ∞ and null heads of paths by $*$. The rows in the distance table correspond to destinations, and columns to neighbors.

The *routing vector* is a message sent by any node i to one of its neighbors for iteration n, and is denoted by $V^{i,j}(n)$. It is a vector of entries, each of which consists of a destination node x, a distance to the destination x from the sending node i, denoted by $D_{ix}(n)$, and the head of the path whose distance is $D_{ix}(n)$ from node i to node x, denoted as $h_{ix}(n)$. Note that the routing vector contains the iteration number n. The size of the distance-table and routing vector depends on the number of functioning nodes and links.

The principal features of the new distributed algorithm are briefly described in the following.

Every node participating in K-DPA is initialized by the local information for the initial graph G^1. The weight $w(i,j)^n$ of any link (i,j) is initialized as ∞ for $1 < n \leq K$. We associate with each iteration a CKG algorithm. Because of the CKG algorithm, any node

i connected to the sink t will have $RDIST_{it}(1)$ as a finite value at some moment. Suppose that node i is not the source node. (The case that node i is the source node will discussed in next paragraph). At this time, node i simply assumes that this distance is the shortest distance from it to sink t, and do graph transformation necessary for G^2 by setting the weight $w(i,j)^2$ for each adjacent link (i,j), with $D^j_{it}(1)$ finite, as $D^j_{it}(1) - RDIST_{it}(1)$. After this, node i can start the computation for the 2^{nd} iteration by treating newly determined link weights for G^2 as detecting recoveries in the 2^{nd} iteration, specifically link weight changes from ∞ to a finite value. Since the routing table distance to t at each node i can vary time to time, due to the nature of the CKG algorithm for each iteration, node i does the same updating as above for $w(i,j)^2$ by assigning the value $D^j_{it}(1) - RDIST_{it}(1)$, whenever $D^j_{it}(1)$ or $RDIST_{it}(1)$ is changed. It is also possible for $RDIST_{it}(1)$ to change from a finite value back to a infinite value. In this case, the weight $w(i,j)^2$ in each of node i's adjacent link (i,j) us set to be ∞. This can be treated in 2^{nd} iteration as detecting failures on all i's adjacent links.

When the source node s obtains a finite routing table distance with preferred neighbor b, it updates $w(s,b')^2$ as $D^{b'}_{st} - RDIST_{st}(1)$ according to (a) in section 2 where $b' \neq b$ and $D^{b'}_{st}$ is finite, marks $w(s,b)^2$ as ∞ according to (b) in section 2, and sends $HOOK(1)$ message to neighbor b. The $HOOK(1)$ message travels on the currently known shortest path from s to t to make the nodes in this shortest path carry out the graph transformation on the two adjacent links on this path according to (b). When a node u receives a $HOOK(1)$ message from a neighbor v, it marks v as n-predecessor, assigns $w(u,v)^2$ the value 0, sends a $HOOK(1)$ message to its preferred neighbor, say y, for sink t in G^1 when u is not t, and sets $w(u,y)^2$ as ∞. At any moment during the algorithm execution, if node u, other than sink t, changes its preferred neighbor for sink t in G^1, and has a neighbor v marked as n-predecessor, then node u sends an $UNHOOK(1)$ message to its original preferred neighbor for sink t, and sends $HOOK(1)$ to the new preferred neighbor. Any node u detecting the failure on link (u,v), or receiving an $UNHOOK(1)$ message from a neighbor, say v, marked as an n-predecessor will then unmark v and send an $UNHOOK(1)$ message to the preferred neighbor for sink t in G^1. Again, changing preferred neighbor for sink t in 1^{st} iteration can only cause the change on link weights for the 2^{nd} iteration to which the CKG algorithm is tolerant.

Therefore, the impact of the 1^{st} iteration on the 2^{nd} iteration is only through link weight changes for the latter. This influence can occur many times because of the dynamics of the distributed algorithm. However, it is automatically accounted for by the CKG algorithm itself. Note that other iterations do not influence the 1st iteration. Therefore, routing tables for G^1 are guaranteed by the CKG algorithm to become stable in a finite time. This implies that after the convergence of the 1st iteration, the 2^{nd} iteration is not influenced by the 1^{st} iteration, and thus its convergence is also assured by the CKG algorithm.

The interaction between iterations 1 and 2 as described above is the same for any iterations n and $n+1$, for $1 \leq n < K$.

Given the above description, it is possible that a node may at some moment have its highest iteration different from that of its neighbor. We complete the description of the K-DPA by explaining the mechanism to handle this situation.

A node i will participate in the n^{th} iteration, $n > 1$, by maintaining its distance and routing tables DT^n_i and RT^n_i, when the shortest distance to sink t in the $n-1^{th}$ $RDIST_{it}(n-1)$ has a finite value. Let k_i denote the highest iteration in which a node i is participating. If any node i updates $RDIST_{it}(n)$ to be ∞, then all the distance and routing tables DT_i^x and RT_i^x, corresponding to $G^x, n < x \leq k_i$, will be purged; then the value of k_i will be updated to be n, and will be sent to all neighbors of node i. We let each node i maintain the k_j value of each neighbor j, whose value is stored in variable k_{ij}. Upon receiving k_j with value n from a neighbor j, any node i updates k_{ij} to be n, and updates its distance and routing tables DT_i^x and RT_i^x with $n < x \leq k_i$ according to the CKG algorithm, as if the link to node j in G^x fails. Then, node i will send to node j the routing vectors, each of which corresponds to routing tables RT_i^m, $n \leq m < k_i$, in the order from n to k_{i-1} sequentially whether the routing tables RT_i^n to $RT_i^{k_i}$ are changed or not. For any distance $D_{jt}(n)$ received by any node i, if $n > k_i$, then node i simply ignores it. As and when paths are found, s can do the exclusive-or operation locally, to find the K-disjoint paths; source routing is appropriate in this context.

A more detailed algorithm description is in Appendix. Note that CKG(m) means operating the CKG algorithm in the transformed graph G^m.

6. PROPERTIES OF ALGORITHM

Theorem 1 *K-DPA terminates in a finite time after the detection of the occurrence of the last topological change.*

Theorem 2 *When K-DPA terminates, for any graph* $G^n, 1 \leq n \leq K$, *the shortest distance maintained by each node to all the other reachable nodes converge to their correct values; and, the distances maintained by each node to all the other unreachable nodes are marked as* ∞.

Theorem 3 *When K-DPA terminates and follows no topological changes, if source s tries to route to sine t a data packet with a routing stamp corresponding to a path p in* ρ^K, *then this data packet will be routed only on path p and arrive at sink t in a finite time.*

(The proofs are omitted here.)

7. CONCLUSION

We have presented a distributed algorithm for finding K disjoint paths of minimum total length. This algorithm avoids the undesirable bouncing effect, and the counting-to-infinity behavior that exist in the distributed BF algorithm. The new algorithm finds k disjoint paths in K iterations of a distributed shortest-path algorithm applicable to links of arbitrary cost. At each iteration, the new algorithm applies a graph transformation derived from Suurballe's work.

The time performance is bounded by K times the time needed for finding a single shortest path. It may be difficult to obtain an algorithm with better time performance, unless the problem of finding K disjoint paths with minimum total length can be solved without transforming the problem into one of finding the single shortest path K consecutive times. The local storage capacity required at each node by our algorithm is undesirably large when all source-sink pairs must be considered. Reducing the size of the local storage needed appears to be difficult if Suurballe's algorithm is to be applied, because the shortest path reversing step is not avoidable to generate the graph for the computation of the next iteration. Nevertheless, our new algorithm can be applied to a number of practical problems. One example of its applicability is providing a few disjoint paths of minimum total length for each source-sink pair in a relatively small network subject to frequent topological changes. In this case, the storage overhead as associated with our new algorithm is quite manageable, and its performance (in terms of its time and message complexities) is better than the performances of the previous approaches. Another example of applicability of this algorithm is in support of hierarchical routing in large packet-radio network. In this case, our new algorithm can be used to maintain disjoint paths among clusters of nodes to reduce the impact of resource failures on the availability of inter-cluster routes. Because the number of clusters is meant to be much smaller than the number of nodes, the storage overhead of the new algorithm is not a major problem.

Appendix

Algorithm K-DPA (distributed K-Disjoint Path Algorithm)
at any node i

```
Procedure 1 Upon receiving a routing vector $V^{j,i}(m)$
begin
            if $m < k_i$
            then begin
(1)            if $D_{jt} = \infty$
               then begin
                  for $n = m$ to $k_i$ do
                  begin
                     $w^n(i,j) \leftarrow \infty$;
                     call $CKG(n)$ assuming link $(i,j)$ in graph $G^n$ fails;
                     if $CKG(n)$ does not send to $j$ then
                     begin
                           $V^{i,j}(n) \leftarrow RT_i^n$; send $V^{i,j}(n)$ to node $j$
                     end
                  end
               end;
               else begin
                  if $D_{jt}(m)$ is in $V^{j,i}$ then begin
                     $w \leftarrow w^{m-1}(i,j) + D_{jt}(m) - RDIST_{it}(m)$;
                     call CKG($m$) as the weight of link $(i,j)$ is changed to $w$;
                     call CKG($m$) as receiving $V^{j,i}(m)$
                  end
               end;
(2)            if there is an $n$ such that $RDIST_{ij}(n) = \infty, 1 \leq n < k_i$
               then begin
                  $r \leftarrow min\{n | RDIST_{ij}(n) = \infty, n \neq k_i\}$
                  delete $RT_i^n$ and $DT_i^n$ for all $n$ such that $r < n \leq k_i$; $k_i \leftarrow r$;
                  send $V^{i,b}$ to each neighbor with $D_{it}(r) = \infty$
               end;
(3)            if $RDIST_{ij}(k_i)$ is with a finite value and $k_i \leq K$
               then begin
                  $k_i \leftarrow k_i + 1$;
                  generate $DT_i^{k_i}$ and $RT_i^{k_i}$ by letting
                     $w^{k_i}(i,b) = w^{k_i-1}(i,b) + D_{bt}^{k_i-1} - D_{it}^{k_i-1}$;
            call CKG($k_i$) as the adjacent links recover with weights $w^{k_i}(i,b)$
        end;
(4)     if $k_i$ has been changed in steps (1) to (3)
        then send $k_i$ to all neighbors;
(5)     for $n = 1$ to $k_i$ do
        begin
            if $p_{it}(n)$ has changed from a neighbor $u$ to another neighbor $v$
                in steps (1) to (3) and there is a neighbor marked as $n$-predecessor
            then send $UNHOOK(n)$ to $u$, and send $HOOK(n)$ to $v$
        end
end
Procedure 2 Upon receiving $HOOK(m)$ from neighbor $j$
    begin
        mark $j$ as $m$-predecessor; $w^m(i,j) \leftarrow 0$;
        $w^m(i,p^n_{it}) \leftarrow \infty$; send $HOOK(m)$ to neighbor $p^n_{it}$
    end
Procedure 3 Upon receiving $UNHOOK$ from neighbor $j$
    begin
        unmark $j$; $w^m(i,j) \leftarrow D^j_{it}(m) - RDIST_{it}(m)$; $x \leftarrow p^n_{it}$
        $w^m(i,x) \leftarrow D^j_{ix}(m) - RDIST_{it}(m)$; send $UNHOOK(m)$ to neighbor $p^n_{it}$
    end
Procedure 4 Upon receiving link failure of link $(i,j)$
    begin
        for $n = 1$ to $k_i$ do
        begin
            if $j$ is marked as $n$-predecessor
            then send $UNHOOK(n)$ to $p_{it}(n)$; $w^n(i,j) \leftarrow \infty$;
            call CKG($n$) as link $(i,j)$ in graph $G^n$ fails
        end;
        do step (2) of Procedure 1
    end
Procedure 5 upon detecting link recovery of link $(i,j)$ with weight $w(i,j)$
    begin
        call CKG(1) as link $(i,j)$ in graph $G^1$ recovers;
        for $n = 2$ to $k_i$ do
        begin
            $w^n(i,j) \leftarrow D^j_{it}(n) - RDIST_{it}(n)$;
            call CKG($n$) as link $(i,j)$ in graph $G^n$ recovers with weight $w^n(i,j)$;
        end;
        do step (3) of Procedure 1;
    end
Procedure 6 Upon receiving $k_j$ from neighbor $j$
    begin
        $k_{ij} \leftarrow k_j$; if $k_{ij} > k_j$ then
            for $n = k_i$ to $k_{ij}$ do send $RT^n_i$ to $j$.
    end
```

CKG Algorithm (sketch); (node i)

```
Upon receiving vector on link $(i, k)$
(1)     copy each entry into relevant column in $DT$;
(2)     if $RT$ needs change (*if row min change*)
        then {call procedure $RT\_UPDATE$;
            send routing vectors only to neighbors not in-path}
Upon receiving notification of failure $(i, k)$/weight change
        {Delete column $k$; step2}
Upon receiving notification of recover $(i, k, d)$
        {insert column $k$ in $DT$; step2}

Procedure $RT\_UPDATE$;
begin
        initialize all destinations except $i$ to be unmarked
        initialize $i$ to be marked
        for any unmarked destination $j$ do
        begin
            if there is no finite distance in row $j$
            then mark $j$ as undetermined
            else begin
                $TV \leftarrow 0$
                if $D^k_{ij}$ is minimum in row $j$ where $P_{ij} = k$
                then pick $k$: $b \leftarrow k$
                else pick any $b$ such that $D^b_{ij}$ is minimum in row $j$
(3)             $c \leftarrow h^b_{ij}, TV \leftarrow TV \cup \{c\}$
                repeat $c \leftarrow h^b_{ic}, TV \leftarrow TV \cup \{c\}$
                until$D^b_{ic}$ is not minimum of row $c$ or $c$ is marked
                if $c$ is marked as undetermined or
                    $D^b_{ic}$ is not minimum of row $c$
                then mark each node in $TV$ as undetermined
                else if $P_{ic} \neq b$
                    then pick up another minimum and goto (3)
                            or mark each node in $TV$ as undetermined
                            when all minimums have been checked
                    else {mark each node $j$ in $TV$ as determined
                            $RDIST_i(j) \leftarrow D^b_{ij}; HEAD_i(j) \leftarrow h^b_{ij}; P_{ij} \leftarrow b$}
            end
        end
        copying routing table to $V^i$
end
```

References

[1] C. Cheng, R. Reily, S.P.R. Kumar, and J.J. Garcia-Luna-Aceves, "A Loop-Free Extended Bellman-Ford Routing Protocol Without Bouncing Effect," *ACM SIGCOMM Symp. Commun. Arch. and Protocols,* Austin, TX, Sep. 1989.

[2] J.J. Garcia-Luna-Aceves, "A Minimum-Hop Routing Algorithm Based On Distributed Information," *Computer Networks and ISDN Systems,* vol.16, pp.367-382, May 1989.

[3] Jacob Hagouel, "Issues in Routing for Large and Dynamic Networks," *IBM Research Report* RC 9942 (No. 44055) Communications, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, Apr. 1983.

[4] Pierre A. Humblet, "An Adaptive Distributed Dijstra Shortest Path Algorithm," an unpublished paper, Feb. 1989.

[5] J.M.Jaffe and F.M. Moss, "A Responsive Routing Algorithm for Computer Networks," *IEEE Trans. Comm.*, vol.COM-30, no.7, pp.1758-1762, Jul.1982.

[6] P.M. Merlin and A. Segall, "A Failsafe Distributed Routing Protocol," *IEEE Trans. Comm.,* vol.COM-27, pp.1280-1288, Sep. 1979.

[7] Richard Ogier and Nacham Shacham, "A Distributed Algorithm for Finding Shortest Pairs of Disjoint Paths," *IEEE INFOCOM*, pp.173-182, 1989.

[8] B. Rajagopolan and M. Faiman, "A New Responsive Distributed Shortest Path Routing Algorithm," *ACM SIGCOMM Symp. Commun. Arch. and Protocols,* Austin, TX, Sep. 1989.

[9] J.W. Suurballe, "Disjoint Paths in a Network," *Networks*, vol.4, pp.125-145, 1974.

[10] J.W. Suurballe and R.E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, vol.14, pp.325-336, 1984.

[11] W.D. Tajibnapis, "A Correctness Proof of a Topology Information Maintenance Protocol for a Distributed Computer Network," *Comm. of ACM*, vol.20, pp.477-485, 1977.

Maximizing Congestion-Free Message Patterns for Bundled Omega Networks

P.J. Bernhard
Department of Computer Science
Clemson University
Clemson, S.C. 29634

D.J. Rosenkrantz[1]
Department of Computer Science
State University of New York at Albany
Albany, N.Y. 12222

Abstract

We consider a strategy for dealing with congestion in bundled omega networks. Specifically, we consider the problem of constructing a maximum size congestion-free set of messages from a given set of messages. We show that regardless of the bundle size of the network this problem is NP-complete.

1 Introduction

An integral part of any multiprocessor system is the *interconnection network.* Depending on the particular multiprocessor, the interconnection network may provide the means for communication between processors, or between processors and memories. A variety of interconnection networks have been proposed in the literature. In this paper we focus on one particular multistage interconnection network called the omega network [La75]. Networks such as the omega network have been used in several multiprocessors, including the University of Illinois Cedar multiprocessor [Ga83].

The omega network is often referred to as a *unique path* network, since any non-broadcast message to be sent through the network involves a unique path between a source and destination. Thus for a given set of messages it may not be possible to simultaneously transmit the messages through the network because some of the messages may *conflict* with one another, in the sense that they require use of the same output of some switch in the network. In such a case we say that the set of messages contains *congestion.* In order to deal with congestion in the omega network a variety of *message routing strategies* have been developed. Some of these make use of message partitioning [A83], others make use of multiple iterations or "passes" through the network [P80], and still others make use of randomization [MC87]. More recent advances in message routing in other networks are given in [A90].

The first type of routing strategy mentioned above involves partitioning a set S of conflicting messages into some number of subsets $S_0, S_1, ..., S_{k-1}$, called *rounds,* such that each subset is congestion-free. The messages are then transmitted through the network by successively transmitting the messages in each subset. This strategy has been considered in the literature. In [A83] it is shown how a lower bound on the number of rounds can be established for a given set of messages. It is also shown there that $O(\sqrt{N})$ is an upper bound on the number of rounds required by any message pattern that forms a permutation on an N-input, N-output omega network. For the class of bundled omega networks partitioning into a *minimum* number of rounds has been shown to be NP-hard [BR87]. In [WF80, DF88] heuristics for minimizing the number of rounds are presented, and in [RV86] and [BR88] special classes of message patterns are identified for which minimizing the number of rounds can be done in polynomial time.

Although the partitioning strategy has been considered in the literature, previous research has been based on the observation that the time required to transmit a collection of rounds depends on the number of rounds in the partition. Consequently, previous approaches have focussed only on *minimizing* the number of rounds for a given set of messages. In this paper we consider a particular method for partitioning a set of messages into rounds which, in contrast to previous approaches, does not focus on minimizing the number of rounds. Instead, when partitioning a message set, the strategy presented here attempts to *maximize* the number of messages that occur in subsets transmitted first, thereby transmitting as many messages through the network as soon as possible. In other words, given an arbitrary set of messages

---

[1] Research supported in part by NSF Grant CCR 88-03278.

S, we would like to construct a congestion-free subset of the messages that contains as many messages from S as possible. Once such a subset $S' \subseteq S$ has been constructed, the messages in the subset S' are allowed to pass through the network. This process is then repeated on the remaining messages, until all messages have been transmitted.

Each phase of the above process leads to an optimization problem that we call the *maximum independent message set problem*, which consists of selecting a maximum size congestion-free subset of messages from a given set of messages. To the best of our knowledge this problem has not previously been considered in the context of multistage interconnection networks. This is surprising since it is a very natural problem. As the name suggests, the independent message set problem has a natural correspondence with the independent vertex set problem for graphs, which will be discussed in Section 2.

In addition, we consider another approach for dealing with congestion. This second approach is a hardware technique that can be used in conjunction with partitioning. Specifically, we consider a class of networks called *bundled* or *dilated omega networks,* where interconnection links in the omega network are replaced by bundles of wires. An omega network is said to have *bundle size b* if each link in the network consists of b wires. The main advantage of bundling is that up to b signals can pass through each bundle in bit serial mode, one signal per wire. The standard omega network is a bundled omega network with $b = 1$. This class of networks has previously been considered in the literature [BR87, K88]. As we shall show, replacing links with bundles does circumvent the problem of congestion for some message patterns. However, when the combined strategy of partitioning and bundling is considered we show that bundling does not make constructing a maximum size congestion-free set of messages any easier for arbitrary message patterns. Finally, we have omitted a detailed description of the omega network from this paper due to space limitations. The interested reader is referred to [La75] for such a description.

2 Maximum Independent Message Sets

As described above, each message to be sent through an omega network makes use of a unique path through the network. Consequently, a message can be represented by a source-address, destination-address pair, abbreviated as an *(s,d)-pair,* where the source s is the address of an input to the network, and the destination d is the address of an output from the network. In light of this, we can define a *message pattern* to be a set S of (s,d)-pairs. Each (s,d)-pair in S corresponds to a message that is to be sent through the network from some input to some output. Note that this definition imposes no restrictions on what type of message pattern S represents. For example, many (s,d)-pairs may have the same source or the same destination, and any number of communication conflicts may exist in S.

As the name suggests, the maximum independent message set problem has a natural graph theoretic counterpart. Specifically, given an arbitrary message pattern, a *conflict graph* can be constructed that has a vertex for each message in the set, and an edge between each pair of vertices when those vertices correspond to conflicting messages. Similarly, a maximum independent set of vertices for the conflict graph (a maximum size set of vertices that have no edges between them), corresponds to a maximum size, congestion-free set of messages. Thus, the problem of constructing a maximum size, congestion-free set of messages can be directly reduced to the maximum independent vertex set problem for graphs, which is NP-complete [GJ79]. Although this reduction does suggest a relation between the two problems, it only holds for omega networks with bundle size $b = 1$. In addition, the reduction does not prove that the independent message set problem is computationally as hard as the independent vertex set problem for graphs. Specifically, it does *not* show that the maximum independent message set problem is NP-hard. More formally, we now define a problem called *k,b-Independent Message Set,* abbreviated *k,b-IMS,* as follows:

INSTANCE: Set S of (s,d)-pairs, positive integers b and k, where $b, k \leq |S|$.
QUESTION: Is there a subset $S' \subseteq S$ such that $|S'| \geq k$, and S' is congestion-free on an omega network with bundle size b?

Theorem. *k,b-IMS is NP-complete, for any fixed $b \geq 1$.*
Proof. We refer the interested reader to [BR90].

This result suggests that there is a need to understand the structural properties of conflict graphs corresponding to various classes of message sets. In particular, correlating properties of message sets with properties of their conflict graphs can help identify special classes of message patterns for which the *IMS* problem is solvable in polynomial time. This in turn may suggest how to devise message sets for parallel algorithms, as well as how to associate the abstract entities in an algorithm to the actual processors of a multiprocessor, so as to maximize the number of messages in a round, or to minimize the total number of rounds.

References

[A83] Agrawal, D.P., "Graph Theoretical Analysis and Design of Multistate Interconnection Networks," IEEE Trans. Comput., vol. C-32, no. 7,July 1983, pp. 637-648.

[A90] Arora,S., Leighton,T., and Maggs,B., "On-line Algorithms for Path Selection in a Nonblocking Network," *Proc. of the Twenty Second Annual ACM Symposium on the Theory of Computing,* Baltimore, Maryland, May 1990, pp. 149-158.

[BR87] Bernhard,P.J. and Rosenkrantz,D.J., "The Complexity of Routing Through an Omega Network," *Proc. Twenty-Fifth Annual Allerton Conf. on Communication, Control and Computing,* Sept. 1987, pp. 1027-1036. Also appears as a technical report: Comput. Sci. Dept., SUNY Albany, Albany, N.Y., TR 87-12, 1987.

[BR88] Bernhard, P.J. and Rosenkrantz, D.J. "An Efficient Method for the Representation and Transmission of Message Patterns on Multistage Interconnection Networks," *Proc. of the 1988 Symposium on the Frontiers of Massively Parallel Computation,* George Mason Univ., Fairfax, Virginia, also to appear in Journal of Parallel and Distributed Computing.

[BR90] Bernhard,P.J. and Rosenkrantz, D.J. "Maximizing Congestion-Free Message Patterns for Bundled Omega Networks," tech. report: Comput. Sci. Dept., Clemson University, Clemson S.C., 1990.

[DF88] Deogun,J.S. and Fang,Z., "A Heuristic Algorithm for Conflict Resolution Problem in Multistage Interconnection Networks," *Proc. of the 1988 Int. Conf. on Parallel Processing,* Penn. State Univ., pp. 475-478.

[Ga83] Gajski,D., Kuck,D., Lawrie,D. and Sameh,A., "Cedar-A Large Scale Multiprocessor," Proc. 1983 Inter. Conf. on Parallel Processing, Aug. 1983, pp. 524-529.

[GJ79] Garey,M.R. and Johnson,D.S., "Computers and Intractability: A Guide to the Theory of NP-Completeness," W.H. Freeman and Co., 1979.

[K88] Koch,R.R. "Increasing the Size of a Network by a Constant Factor Can Increase Performance by More Than a Constant Factor," Proc. of the 29th *Annual Symposium on Foundations of Computer Science,* Oct. 1988, White Plains, N.Y., pp. 221-230.

[La75] Lawrie,D.H., "Access and Alignment of Data in an Array Processor," IEEE Trans. Comput., vol. C-21, no. 12, Dec. 1975, pp. 1145-1155.

[MC87] Mitra,D and Cieslak,R.A., "Randomized Parallel Communications on an Extension of the Omega Network," J.ACM, Vol. 34, No. 4, Oct. 1987, pp. 802-824.

[P80] Parker, D.S., "Notes on Shuffle/Exchange-Type Switching Networks," IEEE Trans. Comput., vol. C-29, no. 3, March 1980, pp. 213-222.

[RV86] Raghavendra,C.S. and Varma,A.,"Fault-Tolerant Multiprocessors with Redundant-Path Interconnection Networks," IEEE Trans. Comput., vol. C-35, No. 4, April 1986, pp. 307-316.

[WF80] Wu,C.L., Feng,T.Y., "On a Class of Multistage Interconnection Networks," IEEE Trans. Comput., vol. C-29, no. 8, August 1980, pp.694-702.

An Improved Deferred Data Structure for Membership Queries

Anil Bhansali
Steven S. Skiena

Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400

Abstract

Perhaps the best reason to sort a set of keys is to permit fast membership queries over it. Enough queries must be performed to justify the cost of sorting, which is impossible to judge in applications where the number of queries is not known in advance.

We give an algorithm, *lazy quicksort*, which uses the partition tree of conventional quicksort as a search tree and a combination of binary and sequential search to locate the appropriate key. Partitioning occurs during the sequential search, leading to a self-organizing data structure. To implement this algorithm efficiently, we develop a new data structure, the *balanced partition tree*, which has a worst-case $O(\log w_t/w)$ search time and $O(\log w_t)$ update time, and an expected $O(1)$ update time, where w_t is the total number of keys in the data structure and w the number of keys in the subtree headed by the given node. We present analytical and empirical results to show that lazy quicksort is a practical algorithm in that its worst case performance penalty is minimal and its performance superior to previous deferred data structures [KMR88].

Keywords: lazy quicksort, deferred data structuring, sorting, self-organizing data structures, balanced partition trees.

1 Introduction

Sorting is undoubtedly the most common operation performed by computers [Knu73b]. Perhaps the best reason to sort a set of items is to speed up membership searches for particular keys. In case of $r = \Omega(n)$ queries, where n is the number of keys, sorting followed by binary search takes $O((n + r)\log n)$, which is optimal. If only $r \ll n$ queries are made, the cost of sorting is not justified and linear search is optimal. Either of these techniques can be inefficient when the number of queries is not known in advance.

Karp, Motwani, and Raghavan [KMR88] considered this problem in a general study of deferred data structures. They prove that $\Theta((n+r)\log n)$ comparisons are necessary and sufficient for the membership query problem. The lower bound comes from an adversarial argument, and the upper bound is realized by two different algorithms, one based on median sorting and the other on mergesort. The median sorting algorithm builds a binary search tree in a query driven fashion. The algorithm uses a linear time median finding algorithm recursively to build a path from root to leaf in the binary search tree. The total time to answer the first query is given by the recurrence $t(n) = t(n/2) + O(n)$, which is linear. On a subsequent query the partial binary search tree can be used to efficiently find the appropriate unsorted (unexpanded) section of the array, which is then expanded using the same recursive median finding algorithm. A complete binary search tree is created when n queries that visit all n keys have been answered.

Smid, Ching and Mehlhorn [SCM90] show how to process the dynamic membership query problem, where we allow insertion and deletion of keys, in time $O(r \log(n+r) + (n+r) \log r)$ which is optimal.

Unfortunately, finding the median, while linear, is expensive in that the multiplicative constant is relatively high. The best known algorithm for computing the median [SPP81] takes $3n + o(n)$ comparisons and is quite complicated. Thus the first query itself in the recursive median finding algorithm will require $\approx 6n$ comparisons, the expected cost of the first 12 linear searches. To make deferred data structuring a practical technique, these constants must be reduced. In [KMR88] they recognize this problem and give a randomized algorithm with lower expected constants by taking a random sample of $O(n^{5/6})$ elements and using the median of the sample as an approximate median.

The most generally successful sorting algorithm, quicksort [Hoa62], recursively partitions the elements, but spends little effort in assuring that the pivot element is near the median. It is natural to generalize this algorithm to the deferred data structuring paradigm. In this paper we provide an algorithm, *lazy quicksort*, whose performance is better than the algorithm based on recursive median finding [KMR88]. Figure 1 illustrates the tradeoff between the number of queries and the total number of comparisons for implementations of various membership testing algorithms based on experiments conducted with a random permutation of 2000 keys and uniformly distributed queries. Curve Q illustrates the total cost of sorting followed by successive binary searches. Although each query takes only $O(\log n)$ comparisons the initial preprocessing is very expensive. The performance of the median sorting algorithm of [KMR88], as illustrated by curve KMR, shows no initial cost but the total number of comparisons increases sharply before stabilizing. The region over which $KMR < Q$ is small enough to be invisible. The performance of the algorithms described in this paper are illustrated by curves LQ and BP. Not only is the total cost well below KMR, but the number of queries until Q and LQ intersect is much larger than the intersection point between Q and KMR. This intersection point provides the justification for using a deferred data structure, and the difference in the points of intersection of KMR and LQ with Q provides the justification for the use of our algorithm. Note that after a sufficient number of queries the keys get sorted and the curves KMR, LQ and BP become parallel to Q. The reason being that after the keys get sorted, we do a binary search in all cases.

For the partition tree created by *lazy quicksort* to serve as an effective search tree it must be balanced. We define a new class of dynamic weighted binary search trees, *balanced partition trees* and show that the search time is $O(\log w_t/w)$ and the update cost is $O(\log w_t)$ in the worst case, with an expected update time of $O(1)$, where w_t is the weight of the tree and w is the weight of the subtree headed by the key being searched. Balanced partition trees have the property that rebalancing operations induced during update operations progress towards the leaves instead of towards the root. Since most of the operations restructuring/rebalancing takes place near the leaves, the number of rotations required to update/rebalance the tree is minimized. We present analytical and empirical results to show that the performance penalty of lazy quicksort over standard quicksort and binary search is minimal, which makes it quite attractive in practice.

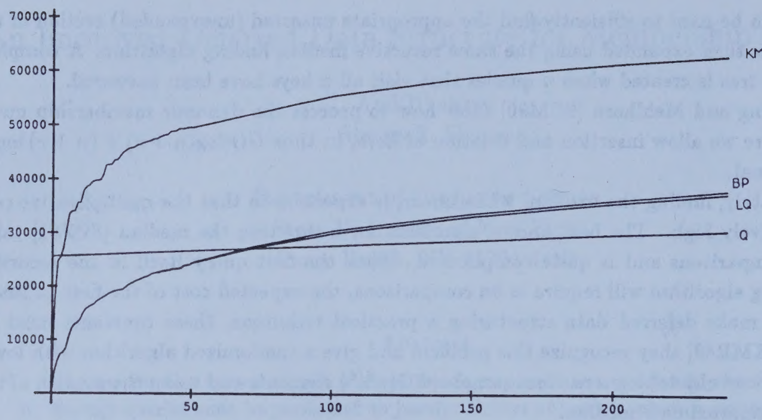


Figure 1: Number of Comparisons vs. Number of queries.

2 Lazy Quicksort

Quicksort is considered to be the best general purpose sorting algorithm. It works by recursively partitioning the data into two sets, based on whether the keys are greater or less than a given partition element. From the cardinality of the two sets, the final position of the partition element can be determined.

Lazy quicksort is based on the observation that the complete partition tree for quicksort is a random binary search tree for the keys [Sed77], with an $O(\log n)$ expected access time for each key. Thus, if explicitly maintained, the partition tree can serve as a search tree.

The algorithm proceeds in a similar fashion as the median sorting algorithm in [KMR88]. For each query, the partition tree is searched from the root. If the item queried was a previous partition element, it will be encountered while searching the tree and is returned. If not, we will reach a leaf in the partition tree, describing a section of the array which has not yet been partitioned. This section of the array is now partitioned. The search for the key is performed during the partitioning phase. Therefore on each query, which does not query some prior partition element, we perform one complete partition step in the course of finding the key.

At each node in the partition tree, we keep track of whether the given subsection of the array is fully sorted. If at any stage the left and right subsections defined by the children of the node are fully sorted, we can merge the children and make the node a leaf, thereby reducing the size of the tree. Within a sorted subsection of the array, binary search can be performed instead of a linear search. After a sufficient number of queries, when every key has been visited at least once, the tree will collapse into one node. Each subsequent query can now be answered by a simple binary search in $O(\log n)$ time.

A problem with this scheme is that, in the worst case, the partition tree can become very skewed, degenerating to linear search. This can be eliminated by maintaining a balanced partition tree. Although height balanced data structures like AVL trees [AL62] and (a,b) trees [Meh84], and static weight balanced trees such as BB[α] trees [NR73] achieve logarithmic access time, they do not support weighted data. Since the access frequency of a node in the partition tree is a function of the number of keys below the node, for best performance our data structure must be a dynamic weight balanced tree. However, since the weights of each node can be easily determined, the use of splay trees [ST83] is inappropriate. In addition, whereas splay trees have an amortized $O(\log n)$ access/update time, we will show that balanced partition trees achieve an absolute $O(\log n)$ access time. Further, we will show that the expected number of rotations to rebalance a balanced partition tree is constant. Although D-trees [Meh84] and weighted BB[α] trees achieve the same absolute time bounds as our BP[α] trees, they require much more complicated restructuring operations.

3 Balanced Partition Trees

We now define a new class of weighted binary search trees called *balanced partition trees* or *BP[α] trees*, $1/2 \leq \alpha < 1$. Associated with each leaf l of the tree is a contiguous range of s keys and weight $W(l) = s$. Each internal node M, of the tree is associated with one key and a weight $W(M) = W(M_l) + W(M_r) + 1$, where M_l and M_r denote left and right child of M respectively. The *root balance* of a node M is given by $Rb(M) = W(M_l)/W(M)$. An edge (X, Y) from node X to its child Y, is said to be *heavy* if $W(Y) > \alpha W(X)$. A left-heavy (right-heavy) edge (X, Y) is a heavy edge from X to its left (right) child Y. Non heavy edges are referred to as *light* edges. A node is *right-imbalanced* (*left-imbalanced*) if the edge incident on its right child is right-heavy (left-heavy). A node M is *balanced* if $1 - \alpha \leq Rb(M) \leq \alpha$, *i.e.* if it is neither left-imbalanced or nor right-imbalanced.

A *BP[α] tree* is a binary search tree where no two consecutive edges are both left-heavy or both right-heavy, and no three consecutive edges are heavy.

3.1 Operations on BP[α] trees

Instead of arbitrary insertions and deletions, BP[α] tree modifications will be restricted to splitting (partitioning) the leaves or merging adjacent leaves. An initial BP[α] tree starts out as a single node with n keys and subsequent nodes are added by split operations. The three operations are:

1. **Query** : Is a particular key in the tree?
2. **Split** : Select a key from a leaf and partition the keys of the leaf into the left and right children of the node. Note that on a split the total weight of the tree remains constant. A split is said to be *even* if the new node M generated due to the split operation is balanced, else it is *uneven*. Restructuring may be necessary on an uneven split to maintain the balance in a $BP[\alpha]$ tree.
3. **Merge** : Two leaves l and r with the same parent P are merged to create a new leaf L such that $W(L) = W(r) + W(l) + 1$. The weight of the tree remains constant on a merge operation. Merge operations require no rebalancing to maintain the properties of a BP[α] tree.

BP[α] tree can be restructured after splits by performing rotations. Figure 2 and Figure 3 show the BP[α] tree before and after left-left (LL) and left-right (LR) rotations, respectively. Right-right (RR) and right-left (RL) rotations are analogous. We adhere to the following convention while marking the edges in the tree. All heavy edges which can possibly induce further rebalancing/restructuring operations are marked with thick lines. Edges which can possibly be heavy and can lead to further rotations are marked with dashed lines. All other edges are marked with thin lines.

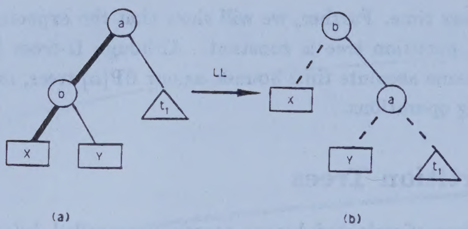


Figure 2: LL Rotation.

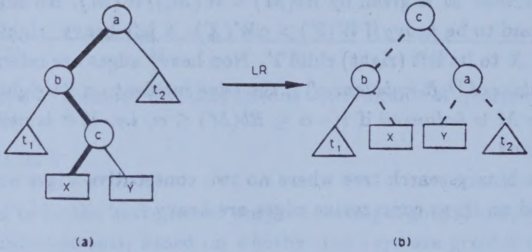


Figure 3: LR Rotation.

3.2 Properties of BP[α] trees

As mentioned earlier an uneven split operation may require restructuring (rotations) operations on the BP[α] trees to restore the balance. Lemma 1 and Theorem 2 are important in bounding the number of rotations required to rebalance a BP[α] tree after a split.

Lemma 1 *In any BP[α] tree with $\alpha \geq \phi$, where $\phi = (\sqrt{5}-1)/2$, the node about which an LL (RR) rotation takes place cannot be right (left) imbalanced after the rotation, and in case of an LR and RL rotation at level l, if the descendant (involved in the rotation) at level $l+2$ is right (left) imbalanced, the node about which the rotation takes place cannot be left (right) imbalanced after the rotation.*

Proof: This can be shown by considering the possible weight distributions on the nodes involved in the rotation. Details can be found in [BS89]. ▮

Theorem 2 *Any split in a BP[α] tree T, with $\alpha \geq \phi$, where $\phi = (\sqrt{5}-1)/2$, requires at most one rotation per level to rebalance the tree.*

Proof: The proof follows from an exhaustive case analysis [BS89] ▮

To find the number of comparisons required to find a particular key in the balanced partition tree we need to find the depth of a node m with weight W_m.

Theorem 3 *In a BP[α] tree T with weight W_T, the depth of a node m with weight W_m is at most*

$$\frac{3\log_2(W_T/W_m)}{\log_2\alpha}$$

Proof: This follows directly from the definition of BP[α] trees. Given any three consecutive nodes on a path from the root to any leaf, at least one of them must be balanced. Therefore on any path of length three we encounter at least one light edge. This implies we prune at least $(1-\alpha)$th of the weight of any node on traversing any three consecutive edges from that node towards any leaf. Thus in the worst case,

$$W_m \leq W_T(\alpha)^{\frac{k}{3}}$$

Solving for k gives the maximum height of node m. When $\alpha = \phi$, we have $k \approx 4.321 \log_2(W_T/W_m)$ ∎

Corollary 4 *$O(\log(W_T/W_m))$ comparisons are sufficient to find a key in an internal node m in a BP[α] tree with weight W_T. If the key is in a leaf, $O(\log(W_T/W_m)+k)$ comparisons are sufficient, where k is the number of keys in the leaf node m.*

Proof: Follows directly from Theorem 3 and the fact that a linear search of at most k keys is required to find the key at the leaf level. ∎

We can now provide a bound on the total number of splits required to restore the balance in a balanced partition tree after an uneven split.

Theorem 5 *In a BP[α] tree of height h, at most $h-l+2$ rotations are required to rebalance a tree after a split operation at level l.*

Proof: From Theorem 2 it follows that one rotation per level is sufficient to rebalance a BP[α] tree after a split. The ith rotation occurs at least one level higher than the $(i-1)$th rotation. Therefore, if the split occurred at level l the first rotation can occur at level $l-2$ (RL or LR) or at level $l-1$ (LL or RR). Since the tree is of height h the maximum number of rotations required are $h-l+2$. ∎

The actual number of rotations will usually be far less. From the case analysis of Theorem 2 it follows that that the number of heavy edges in the tree before a rotation is less than or equal to the number of heavy edges after rotation. This implies that even though we might add new heavy edges at certain nodes, we delete at least as many heavy edges from other nodes. This also means that each rotation can potentially create a new balanced node but can never cause previously balanced nodes to become unbalanced. Once a node m is balanced there cannot be further rotations with m as the pivot. This implies that if, in the course of rotations filtering downward after a split, a heavy edge is created incident on a balanced node, the sequence of rotations will be terminated. Thus a balanced node acts as a blocking node to further rotations. From the definition of balanced partition trees it follows that any path from root to leaf encounters at least one balanced node among any three consecutive nodes.

Theorem 6 *The expected number of rotations on a split is $O(1)$, when the split point is selected with uniform probability over the keys in each leaf.*

Proof: Let p be the probability that a new heavy edge created by a rotation is incident on a node which can induce further rotations. To generate a sequence of rotations we must encounter such a node after every rotation. Since every internal node in the tree started as the result of a split, the probability of a given node resulting from a balanced split is $2(\phi - 1/2) = \sqrt{5} - 2 \approx 0.2361$.

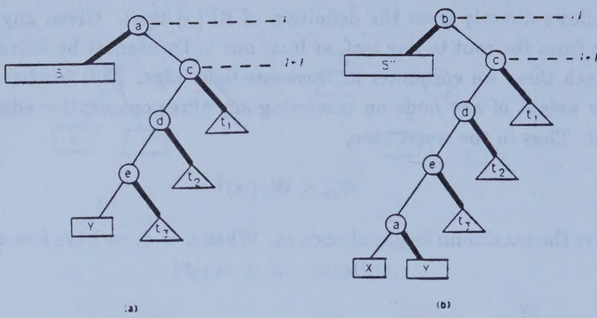


Figure 4: BP Tree to serve as counter-example for amortized analysis.

Thus $p < 1-0.2361$ since blocking nodes cannot induce further rotations. Therefore the expected number of rotations per split is bounded by,

$$S = \sum_{i=1}^{\infty} i(1-p)p^i = \frac{p}{1-p} \approx 3.24$$

∎

The percentage of blocking nodes gives only a crude bound on p, so tighter analysis can lower p and thus the expected number of rotations. We have implemented this algorithm and empirical results suggest the number of rotations per split is approximately 0.15.

The constant bound on the number of rotations per split operation poses the question of an amortized $O(1)$ bound on the number of rotations. Unfortunately, such a bound does not exist. The BP[α] tree in Figure 4a induces $O(\log W_T)$ rotations per split for an arbitrary number of splits. After the first unbalanced split at node S, rotations are induced to restructure the tree. Figure 4b denotes the tree after rebalancing. Note that the structure of the tree after restructuring is similar to the one before. Another unbalanced split occurring at node S' will induce the same sequence of rotations and shows that we cannot have an amortized $O(1)$ bound on the number of rotations.

The merge operation, which collapses two leaves into their parent, can be done in $O(1)$ time since no rotations are ever necessary to rebalance the tree. This is obvious since the weight of the parent node does not change in a merge.

4 Analysis of Lazy Quicksort

The median sorting algorithm of [KMR88] offers asymptotically optimal worst case performance of membership queries. Because quicksort has worst case $O(n^2)$ behavior, no such claims can be made about an algorithm which is based on it. However, we can show that the worst case penalty of using lazy quicksort over conventional quicksort and binary search is minimal:

Theorem 7 *The number of comparisons performed by lazy quicksort with balanced partition trees during r queries over n keys is the minimum of $O(Q + r \log n)$ and $O(rn)$, where Q is the number of comparisons performed by a conventional quicksort of the n keys.*

Proof: In a balanced partition tree, the cost of any query is $O(\log n)$. The comparisons performed in partitioning are exactly the same comparisons as would have been performed in the course of a conventional quicksort, although possibly in a different order. The cost of the sequential search at most equals the cost of the partition. Obviously, no partition requires more than $O(n)$ comparisons, giving the second bound. ∎

If both children of an internal node represent sorted sets of keys, the leaves are merged and the node collapses. Binary search can be performed on sorted leaves. Once the entire tree collapses, the array is sorted and there is no further penalty for having used the deferred data structure. Thus it is of interest to know the expected number of queries until the array is sorted.

Theorem 8 *The expected number of queries required to completely sort n keys is $O(n \log n)$.*

Proof: Note that sorting is complete once each of the n keys have been selected as a partition element. The problem is thus similar to the coupon collector's problem [Knu73a] which asks for the expected number of samples of N items taken uniformly at random such that each item is sampled at least once.

Assume that we have already selected k distinct keys. $E(x_k)$, the expected number of queries to get the $(k+1)$th distinct key is given by,

$$E(x_k) = \sum_{i=1}^{\infty} i(\frac{k}{N})^{i-1}(\frac{N-k}{N}) = \frac{N}{N-k}$$

Therefore the expected number of queries required to select all N keys at least once, $E(N)$, is given by,

$$E(N) = \sum_{i=1}^{N} E(x_i) = \sum_{i=1}^{N} \frac{N}{N-i} \approx N \log_e N$$

∎

5 Empirical Results

We have implemented the lazy quicksort algorithm with and without balanced partition trees and have also implemented the median sorting algorithm of [KMR88], using a randomized median finding algorithm. Simulation runs were carried out on random permutations of 250 to 50000 keys with queries distributed uniformly over the entire range of keys. These results were partially presented in Figure 1.

Table 1 shows the number of comparisons needed to sort the keys by different algorithms. The total number of comparisons until sorting occurs for lazy quicksort with (BP) and without (LQ) balanced partition trees was ≈ 43 % more than that of the static quicksort algorithm (Q), versus 116 % more for median sorting (KMR).

Table 2 gives the equalling query, the number of queries when the total cost of the deferred data structure equals that of the static algorithm. The equalling query appears to grow proportional to the square root of the number of keys for lazy quicksort (BP) as opposed to logarithmically with median sorting (KMR). It is the size of this window which justifies using the self-organizing structure in the first place.

Keys	Q	LQ	BP	KMR
250	2247	3289	3094	3349
500	5250	8590	7450	9549
1000	11261	17004	15921	25403
2000	25561	42776	39376	54513
5000	68687	99185	87963	155494
10000	148426	231985	210918	365172
25000	417977	706336	646061	974530
35000	628017	1070076	898092	1442009
50000	892335	1390678	1229880	2129516

Table 1: No. of Keys vs. Total No. of Comparisons

Keys	KMR	BP
250	3	22
500	4	26
1000	5	46
2000	7	55
5000	8	101
10000	9	127
25000	14	171
35000	17	236
50000	22	318

Table 2: Equalling Query

6 Conclusions

We have shown that lazy quicksort is a practical self-organizing data structure for performing membership queries. Analytical and empirical results show that the total number of comparisons can never be significantly worse than the cost of performing a quicksort. All queries are logarithmic exclusive of sorting cost, and since the expected sorting cost is $O(n \log n)$ and $O(n)$ partitions are performed, this averages to a logarithmic time per query.

Balanced partition trees cannot be generalized to the dynamic situation [SCM90] within the same performance bounds. As in [KMR88], our technique supports only the static case of the membership query problem, where no insertions and deletions are allowed. While the partition tree of quicksort lends itself to the dynamic deferred data structuring model we cannot generalize the balanced partition tree to support arbitrary insertions and/or deletions, without compromising on its performance bounds. One reason for this being that a single insertion or deletion, in the worst case, can lead to $O(\log W)$ edges to become heavy. The rebalancing costs will therefore be much more than in the static case. One problem with deferred data structures is that extra storage is necessary to maintain the partition tree. This suggests exploring the idea of implicit deferred data structures, where no extra storage over that of the keys themselves is used. The recursive median algorithm does lend itself to an implicit implementation, since the position of the median can be calculated and by permuting the first and last elements in the subsection we can flag whether the element in the appropriate position is actually the median. Lazy quicksort can be implemented implicitly by performing a sequence of binary searches to find the partition, but this dooms us to an $O(\log^2 n)$ query algorithm. Munro and Suwanda [MS80] discuss various implicit data structures which might be modified to yield a better data structure for lazy quicksort. Also, we can consider even lazier sorting algorithms, where a full partition step is not necessarily completed on each query. The number of comparisons will be further reduced at the expense of adding complexity to the algorithm.

References

[AL62] G.M. Adel'son-Vel'skii and E. M. Landis. An algorithm for the organization of information. *Dokladi Akademia Nauk SSSR*, 146:263–266, 1962.

[BS89] A. Bhansali and S. S. Skiena. *Balanced Partition Trees and Lazy Quicksort.* Technical Report 89-24, SUNY at Stony Brook, 1989.

[Hoa62] C. A. R. Hoare. Quicksort. *Computer Journal*, 4:10–15, 1962.

[KMR88] R. M. Karp, R. Motwani, and P. Raghavan. Deferred data structuring. *SIAM Journal on Computing*, 1988.

[Knu73a] D. Knuth. *Seminumerical Algorithms.* Volume 2 of *The Art of Computer Programming*, Addison-Wesley, Reading MA, 1973.

[Knu73b] D. Knuth. *Sorting and Searching.* Volume 3 of *The Art of Computer Programming*, Addison-Wesley, Reading MA, 1973.

[Meh84] K. Mehlhorn. *Sorting and Searching.* Volume 1 of *Data Structures and Algorithms*, Springer-Verlag, 1984.

[MS80] J. Ian Munro and Hendra Suwanda. Implicit data structures for fast search and update. *JCSS*, 21:236–250, 1980.

[NR73] J. Nievergelt and E. M. Reingold. Binary search trees of bounded balance. *SIAM Journal on Computing*, 2:33–43, 1973.

[SCM90] M.H.M. Smid, Yu-Tai Ching, and Kurt Mehlhorn. Dynamic deferred data structuring. *Information Processing Letters*, 35:37–40, 1990.

[Sed77] R. Sedgewick. The analysis of quicksort programs. *Acta Informatica*, 7:327–355, 1977.

[SPP81] A. Schöhage, M. Patterson, and N. Pippenger. Finding the median. *Journal of Computer and System Sciences*, 13:184–199, 1981.

[ST83] D. D. Sleator and R.E. Tarjan. Self-adjusting binary trees. *STOC*, 15:235–245, 1983.

TYPICAL BEHAVIOR OF PATRICIA TRIES

Bonita Rais[†]
Department of Computer Science
Purdue University
W. Lafayette, IN 47907
U.S.A

Philippe Jacquet*
INRIA
Rocquencourt
78153 Le Chesnay
FRANCE

Wojciech Szpankowski[§]
Department of Computer Science
Purdue University
W. Lafayette, IN 47907
U.S.A

Digital tries occur in a variety of computer and communication algorithms including symbolic manipulations, compiling, comparison based searching and sorting, digital retrieval techniques, algorithms on strings, file systems, codes and communication protocols. We study the depth of the PATRICIA trie, a variation of the trie in which one-way branches are collapsed into a single node. The PATRICIA trie is built over a V-ary alphabet whose keys are strings from the alphabet. The occurrence of the i-th symbol of the alphabet in a key is given by p_i, $i=1,2,...,V$, and the n keys are independent. This paper establishes the limiting distribution for the depth of keys in a PATRICIA trie.

Theorem 1a. Consider the *asymmetric* model of PATRICIA tries described above. Let the random variable D_n be the depth of a randomly chosen key in a PATRICIA trie holding n keys. Then,

(i) For large n the average ED_n depth of a PATRICIA trie is

$$ED_n = \frac{1}{H}\log n + O(1)$$

and the variance $var\, D_n$ of the depth is

$$var\, D_n = \frac{H^2 - H_2}{H^3}\log n + O(1)$$

where $H = -\sum_{i=1}^{V} p_i \log p_i$ is the entropy of the alphabet, and $H_2 = \sum_{i=1}^{V} p_i \log^2 p_i$.

(ii) The random variable $\left[\frac{D_n - ED_n}{\sqrt{var\, D_n}}\right]$ is asymptotically *normal* with mean zero and variance one, that is,

$$\lim_{n\to\infty} Pr\{D_n \le ED_n + x\sqrt{var\, D_n}\} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}dt,$$

In addition, for all integer m the following convergence in moments holds

$$E\left[\frac{D_n - ED_n}{\sqrt{var\, D_n}}\right]^m \to \begin{cases} \dfrac{m!}{2^{m/2}\,(m/2)!} & m \text{ even} \\ 0 & m \text{ odd} \end{cases}$$

where the RHS of the above presents moments of the normal distribution [1].

[†] This research was in part supported by NSF grant CCR-8900305.

[*] This research was done while the author was visiting Purdue University, and was supported by NATO Collaborative grant 00571/89.

[§] This author's research was supported in part by NATO Collaborative grant 00571/89, and in part by AFOSR grant 90-0107, NSF grant CCR-8900305, and by grant R01 LM05118 from the National Library of Medicine.

Theorem 1b. Consider the *symmetric* model of PATRICIA tries described above. Then,

(i) For large n the average ED_n depth of a PATRICIA trie is [4]

$$ED_n = \log_V n + \frac{\gamma}{\log V} - \log_V(1-1/V) + \frac{1}{2} + P_1(\log n)$$

and the variance $var\, D_n$ of the depth is constant, more precisely [4]

$$var\, D_n = \frac{\pi^2}{6\log^2 V} + \frac{1}{12} - \frac{2}{\log V}\log\prod_{j=1}^{\infty}\left[1+\frac{1}{V^j}\right] + P_2(\log_V n)$$

where $\gamma = 0.577$ is the Euler constant, and $P_1(\log_V n)$ and $P_2(\log_V n)$ are fluctuating functions of very small amplitude.

(ii) The limiting distribution is periodic , that is:

$$\lim_{n\to\infty} | Pr\{D_n < x + \log_V n\} - \Delta(x, \log n) | = 0$$

where $\Delta(x,y)$ is periodic with respect to y with period $\log V$ (for more details see [3]). The convergence also holds for all moments [2].

Remarks.

Comparison with regular tries. The variance of the depth, as well as the expected depth, for a PATRICIA is smaller than for regular tries. From the above we may conclude that PATRICIA trie is a better balanced trie than the regular trie, and this is discussed next.

How well is the PATRICIA balanced ? A tree built over a V-ary alphabet is well-balanced if (a) the average depth of a key is $\log_V n + O(1)$, (b) the variance of the depth is small, and (c) large deviations from the average value are very unlikely. In the symmetric case, these conditions are clearly met, thus we can expect that the PATRICIA trie is well-balanced. In the asymmetric case, the coefficient $1/H$ in the mean shows that the more asymmetric the distribution of the symbols is, the more skew the PATRICIA trie is. However, since the limiting distribution is normal, we have the following for large n:

$$Pr\{ | D_n - ED_n | > 3\sqrt{var\, D_n} \} = 0.003$$

This means that with very high probability, the depth of a random key is within three standard deviations of the expected depth. But the variance is $O(\log n)$ so its square root is much smaller than $\log n$ (the average depth). In other words, large excursion from the average value happens very rarely, and from Cramer's theorem we note that such a probability decays exponentially fast with the distance from the average value [1]. Thus we can conclude that the PATRICIA is, on average, well-balanced.

Poisson model. In the proof of our theorem, we will also establish similar convergence results for the Poisson model. That is, once normalized and centered, the depth of a key in a PATRICIA trie with Poisson number of records, under an asymmetric model, is asymptotically normal. ■

REFERENCES

1. W. Feller, *An Introduction to Probability Theory and its Applications*, John Wiley & Sons, New York (1988).
2. P. Jacquet and M. Regnier, Limiting Distributions for Trie Parameters, *Lecture Notes in Computer Science*, 214, 196-210 (1986).
3. B. Pittel, Paths in a Random Digital Tree: Limiting Distributions, *Adv. Appl. Prob.*, 18, 139-155 (1986).
4. W. Szpankowski, Patricia Tries Again Revisited, *Journal of the ACM*, (1990), in press.

Parallel Priority Queues

Maria Cristina Pinotti[†] and Geppino Pucci[‡]

[†]Istituto di Elaborazione dell'Informazione, C. N. R., Pisa, Italy.
[‡]Dipartimento di Informatica, Università di Pisa, Pisa, Italy.

Abstract

A Parallel Priority Queue (PPQ) is defined as an abstract data type for storing a set of integer-valued items and providing operations such as insertion of n new items or deletion of the n smallest ones. In this paper, the PPQ data type is implemented on the PRAM model of parallel computation. Two data structures are introduced, the n-Bandwidth-Heap (nH) and the n-Bandwidth-Leftist-Heap (nL), based on the well known sequential binary-heap and leftist-heap organizations, respectively. Efficient parallel algorithms are then given for the basic operations of insertion, deletion and heap construction on nH and nL, based on known parallel sorting and merging algorithms.

Keywords: Data structures, parallel algorithms, analysis of algorithms, heaps, PRAM model.

This work has been partly supported by Ministero della Pubblica Istruzione of Italy and by the C.N.R. project "Sistemi Informatici e Calcolo Parallelo".

1. Introduction

A *Priority Queue* (PQ) is an abstract data type used in a wide variety of algorithms, for storing a set of labelled items and selecting the one associated with the smallest label [1]. In this note, we propose two Parallel implementations of a Priority Queue (PPQ) on the CREW-PRAM. In this model of computation n RAM processors $P_1, \ldots, P_n$ have concurrent unit time access to a shared memory, with the provision that two or more processors may read a cell simultaneously but concurrent write accesses to the same cell are prohibited [4].

We require that a PPQ allow efficient simultaneous insertions of n new items, one for each processor, or the removal of the n items associated with the n smallest labels. A PPQ can be useful for the parallel implementation of those techniques, such as Branch-and-Bound, which imply the solution of several subproblems, each associated with a different cost [5]. By using a PPQ, at each stage the processors may efficiently select the n "more promising" subproblems to be solved.

Formally, a PPQ Q is an abstract data type consisting of a collection of (possibly replicated) items with associated integer-valued labels. Three operations are defined, namely:

1) *Insert* ($<i_1, \ldots, i_n>, Q$) for the insertion of items $i_1, \ldots, i_n$ in Q;

2) *Deletemin (Q)* for the deletion and return of the n smallest labelled items;

3) *Makequeue (S, Q)* for the construction of Q with the items of a set S.

An additional queue operation is provided by *meldable* PPQs, namely:

4) *Meld* (Q_1, Q_2, Q) for the combination of PPQs Q_1 and Q_2 into Q.

In this paper, we devise two PPQ implementations, based on the sequential structures known as *binary heap* [8] and *leftist heap* [6]. The first implementation, called *n-Bandwidth-Heap* (nH), is unmeldable. The second, called *n-Bandwidth-Leftist-Heap* (nL) is instead meldable.

2. Background and definitions

First, let us briefly review some of the properties of the mentioned sequential data structures. Both of them are based on *heap-ordered* binary trees, that is trees where the label of each node is smaller than or equal to the labels of its children. A binary heap is a complete binary tree, where some rightmost nodes of the last level may be absent. Hence, binary heaps keep a logarithmic depth and can be stored in an array, where the positions $2 \cdot i$ and $2 \cdot i+1$ contain the left and right children of node i.

Recall that the *rank* of a node x of a binary tree is defined as:

$$\mathrm{rank}(x) = \begin{cases} 0, & \text{if } x \text{ is void;} \\ 1, & \text{if } x \text{ is a leaf;} \\ 1+\min\{\mathrm{rank}(\mathrm{left}(x)), \mathrm{rank}(\mathrm{right}(x))\}, & \text{otherwise;} \end{cases}$$

where left(x) and right(x) denote the left and right children of x. A leftist heap is a heap-ordered *leftist tree*, that is, a binary tree such that, for each node x, $\mathrm{rank}(\mathrm{left}(x)) \geq \mathrm{rank}(\mathrm{right}(x))$. It can be shown that, in a leftist tree, the shortest path from the root to a leaf is the rightmost one in the tree, and that this path has at most logarithmic length in the number of nodes. Note that, unlike binary heaps, a leftist heap must be stored as a linked structure, with rank explicitly stored with the nodes.

In a parallel context, the above structures seem not to be suitable for an efficient realization of PPQ operations. For instance, using a binary heap or a leftist heap, the n^{th} smallest value, needed by Deletemin, would be available after $\Omega(n)$ time, as the position of the i^{th} smallest item depends on the position of the previous i-1 smallest ones. To find the n smallest elements in parallel, in $O(1)$ time, we will store a set of n items at each node of the structure so that the maximum label in the set is equal to or smaller than all the labels stored at the node's descendants. Under this organization, that we call *extended heap order*, the n smallest values are found in the root.

Formally, we define an *n-Bandwidth-Heap* (nH) and an *n-Bandwidth-Leftist-Heap* (nL) respectively as a binary heap and a leftist heap, whose nodes, containing n items, are arranged in extended heap order. Letting $m = f(n) \cdot n$ be the total number of items stored in the heap, we denote the height of an nH, or the length of the rightmost path of an nL by $h \in O\left(\log\frac{m}{n}\right)$. The bandwidth heaps can be realized like their sequential counterparts, with the only difference that each node contains a vector of n items. For implementation purposes, we require that the items at a node appear in sorted order.

3. PPQ Operations

Let us present the algorithms to realize the PPQ operations on nH and nL. The building blocks of such algorithms are PARALLEL-SORT and PARALLEL-MERGE that are employed to efficiently establish and preserve the extended heap order during the operations. Recall that, with an n-processor CREW-PRAM, n items can be sorted in time [3]:

$$T_{Sort} = \Theta\ (\log n); \tag{1}$$

while merging two vectors of cardinality k_1 and k_2 can be accomplished in time [7]:

$$T_{Merge} = \Theta\ (\min\ \{(k_1+k_2)/n + \log(k_1k_2\log n/n),\ (k_1+k_2)\ (\log\log\ (k_1+k_2))/n\}). \tag{2}$$

As a particular case of relation (2), we have that two vectors of size n can be merged in time O (loglog n) by using n processors, while the same operation can instead be performed in time O (log n) if only n/log n processors are available.

3.1 PPQ operations on nH

Let E_P denote the ordered sequence of n items stored in a node P of an nH and $\max(E_P)$ denote the maximum element of E_P. Moreover, a *path* $\pi = P_1, \ldots, P_k$ is a sequence of nodes of heap Q from the root P_1 to a leaf P_k, and for a given path π, E_π denotes the concatenation $E_{P_1}\, E_{P_2} \ldots E_{P_k}$. Note that E_π is an ordered sequence. However, when we inserte or delete a node, we will replace the set stored in P_1 or P_k with another ordered set, which may violate the overall order of E_π. To re-order E_π after such replacement, we define the following procedure on π:

```
procedure REARRANGE ($\pi$, P $\in$ {$P_1$, $P_k$}):
    begin
        E := PARALLEL-MERGE ($E_P$, $E_{\pi - P}$);
        Let $E^1, \ldots, E^{|\pi|}$ be the $|\pi|$ consecutive subsequences of E with cardinality n;
        for i := 1 to $|\pi|$ do $E_{P_i} := E^i$ endfor;
    end.
```

The time complexity C_R of REARRANGE is dominated by the time of PARALLEL-MERGE between two sequences of length $n(|\pi| - 1)$ and n. We have from (2):

$$C_R \in\ O\ (\min\ \{|\pi|+\log n,\ |\pi|\log\log n.\}). \tag{3}$$

Insertion. To insert n new items into an nH, say Q, we first place them, in sorted order, in the leftmost vacant leaf L of Q; then we rearrange the path π from the root of Q to L with REARRANGE (π, L). It is crucial to note that this operation preserves the extended heap order. In fact, for each node $P_i \in \pi$, $\max(E_{P_i})$ is not incremented by REARRANGE. Then, the minimum elements stored in the children of P_i are both greater than $\max(E_{P_i})$. A simple program for insertion is the following:

```
program INSERT ({i_1, ... ,i_n}, Q):
  begin
    Let L be the leftmost vacant leaf of Q;
    E_L := PARALLEL-SORT ({i_1, ... ,i_n});
    Build the path π from root to L;
    REARRANGE (π, L);
  end.
```

Since $|\pi| = h$, π can be built in time O(h). Hence, the time complexity C_I of INSERT is determined by the initial sorting phase, and by the complexity of REARRANGE. From (1) and (3), it easily follows that:

$$C_I \in O\,(h+\log n). \tag{4}$$

Deletion. Let a *minimum-path* μ of Q be recursively defined as:

1) the root of Q belongs to μ;

2) if a non leaf node P belongs to μ, then X belongs to μ, such that X, Y are the children of P, and $\max(E_X) \leq \max(E_Y)$, or Y is void.

We define a new procedure ADJUST on μ:

```
procedure ADJUST (μ):
  begin
   for each P∈ μ do
     if Brother(P) exists
       then    E := PARALLEL-MERGE (E_P, E_Brother(P));
               Let E^1, E^2 be the left and right halves of E;
               E_P := E^1;
               E_Brother(P) := E^2;
     endif
   endfor
  end.
```

Note that ADJUST does not violate the extended heap order. A straightforward implementation of ADJUST on the PRAM takes time O (h loglog n). However, for

$h > (\log n)/\log\log n$, a better time complexity is achieved by assigning $n/\log n$ processors to each node of μ to perform the PARALLEL-MERGE operation between that node and its brother. The algorithm will then consist of $\lceil h/\log n \rceil$ phases, each one relative to at most log n nodes of μ. Each phase takes O(log n) time (from (2)). Therefore, ADJUST can be realized in time:

$$C_A \in O(\min\{h \log\log n, \max\{h, \log n\}\}). \qquad (5)$$

Let now R and L denote, respectively, the root and the rightmost non vacant leaf of Q. Deletion is performed by the program DELETEMIN reported below. Recalling that the set E_R contains the n smallest elements of Q to be deleted, at first deletion returns E_R, and replaces the set in R with the elements in L. Then ADJUST and REARRANGE are called on the minimum path μ of Q to re-establish the extended heap order. Note that in the ordered sequence E_μ, created by REARRANGE, some elements previously residing on $P_i \in \mu$ may now be assigned to its father P_{i-1}. However, the extended heap order is not violated as ADJUST has modified μ so that $\max(E_{P_i})$ (hence all the values of E_{P_i}) becomes smaller than the minimum value of Brother(P_i).

```
program DELETEMIN (Q):
    begin
        Let R, L be the root and the rightmost non vacant leaf of Q;
        Return $E_R$;
        $E_R := E_L$;
        Build the minimum-path $\mu$;
        ADJUST($\mu$);
        REARRANGE($\mu$, R);
    end.
```

The time complexity C_D of Deletemin is dominated by the execution of REARRANGE and ADJUST, since the other operations take altogether O(h) time. Hence:

$$C_D \in O(\min\{h+\log n, h \log\log n\}). \qquad (6)$$

Note that, for $h \geq \log n$ (i.e., $m \geq n^2$) Deletemin takes time proportional to the nH height. The slowdown, for the case $h < \log n$ is due to the PARALLEL-MERGE algorithm, which cannot reach unit efficiency if the number of processors is “close” to the number of elements to be merged [2].

Construction. Let S be the set of elements to be stored in the new heap Q. We first build an nH tree whose nodes contain sorted items. Such nodes do not necessarily satisfy the extended heap order, which is subsequently created, level by level, starting from the leaves. Specifically, for each subtree T at level k, its minimum-path μ_T is built, adjusted and rearranged. The construction is performed by the following program:

```
program MAKEQUEUE (S, Q):
    begin
      for i:=0 to m/n - 1 do            -- Sorting phase
        S := S - {e_1, ..., e_n};
        Q[i·n+1, . . . , (i+1)·n] := PARALLEL-SORT ({e_1, . . . ,e_n});
      endfor;
      for k := h downto 1 do            -- Heapify phase, from the leaves
       for each P at level k do
          Let T be the nH rooted at P;
          Build the minimum-path μ_T;
          ADJUST (μ_T);
          REARRANGE (μ_T, P);
        endfor;
      endfor;
    end.
```

The correctness of the above algorithm can be easily proved by induction on the height of the nH. As to its time complexity, from (1) it follows that the sorting phase can be accomplished in parallel time O((m/n)log n). During the heapify phase, at each node P at level k, all processors operate in parallel to re-establish the extended heap order in the subtree rooted at P. Hence, each step of the inner loop takes time:

$$O(\min\{k+\log n, k \log\log n\}),$$

for a total time of

$$O\left(\sum_{k=1}^{h} \frac{m}{n2^k} \min\{k+\log n, k \log\log n\}\right) = O((m/n) \log\log n).$$

Therefore, the complexity C_{MQ} of Makequeue is altogether:

$$C_{MQ} \in O((m/n)\log n). \qquad (7)$$

3.2 PPQ operations on nL

Due to their vectorial representation, nH cannot efficiently realize meldable PPQ. Melding is instead the basic operation for nL, since all the other PPQ operations of insertion, deletion, and heap construction can be implemented through melding.

Melding. To meld two nL, Q_1 and Q_2, into a nL Q, we combine their rightmost paths ρ_1 and ρ_2 into a single path ρ of length $h = h_1+h_2$, where $h_1=|\rho_1|$ and $h_2=|\rho_2|$. The remaining nodes of Q_1 and Q_2 are then attached to this path opportunely. Finally rank readjustment is performed, to guarantee the leftist property in Q.

We represent a node P of an nL by a record P = (E, left, right, rank), where P.E is the sorted sequence of n values stored in P (denoted by E_P is section 3.1); P.left and P.right are the pointers to the left and right children of P, respectively; and P.rank is the rank of P. Furthermore, for a node P, we define L(P) as the pair (P.left, max(P.E)). As before, we denote by $E\rho_i$, with i = 1, 2, the concatenation of the P.E stored at the nodes of path ρ_i. Similarly, we define $L\rho_i$ to be the sequence of the pairs relative to the nodes of ρ_i. We say that $L\rho_i$ is *ordered*, because the second components of its pairs appear in non decreasing order. Melding is performed by the program MELD reported below. (Note the use of the Pascal-like notation new(P) and P^, to denote a new record to be created, and its fields).

```
program MELD ($Q_1$, $Q_2$, Q):
  begin
    Let $\rho_1$ and $\rho_2$ be the rightmost paths of $Q_1$ and $Q_2$, respectively; and let
    $h_1 = |\rho_1|$ and $h_2 = |\rho_2|$.
    E := PARALLEL-MERGE ($E\rho_1$, $E\rho_2$);
    L := PARALLEL-MERGE ($L\rho_1$, $L\rho_2$);
    Let $E^1, \ldots, E^{h_1+h_2}$ be the ($h_1+h_2$) consecutive subsequences of E with
    cardinality n;
    Let $L^1, \ldots, L^{h_1+h_2}$ be the ($h_1+h_2$) first components of the pairs in L;
    h := $h_1 + h_2$;
    new(P); P^ := ($E^h$, $L^h$ nil, 1);
    for i := h-1 downto 1 do
      N := ($E^i$, $L^i$, P, 0);
      if (N.right)^.rank > (N.left)^.rank
                   then swap (N.left, N.right) endif;
      N.rank := (N.right)^.rank + 1;
      new(P); P^ := N;
    endfor;
   Q := P;
  end.
```

The merge operation on the sequences $L\rho_1$ and $L\rho_2$ is based on the total order defined over the second field of the pairs. Moreover, note that the **for** loop in MELD performs both the path construction and the rank readjustment phases. To show that the above algorithm is correct, we observe that the construction of the rightmost path ρ guarantees the extended heap order as, for each left subtree T pointed by a node P in ρ, the values in P.E can only be less or equal to the ones in the node of ρ_1 or ρ_2 that originally pointed to T.

Finally, it can be easily seen that the time complexity C_M of MELD is dominated by the PARALLEL–MERGE step needed to create E and L, while all the other phases are performed in O(h) time. From (2), we have:

$$C_M \in O(\min\{h_1+h_2+\log n, (h_1+h_2)\log\log n\}). \tag{8}$$

Hence, for $(h_1+h_2) \in \Omega(\log n)$, the meld operation takes time proportional to the height of the resulting tree.

All the other operations on nL are based on melding. To insert n new items in an nL Q, we make them into a one-node heap and meld it with Q. To delete the n minimal elements from Q, we remove its root and meld the remaining left and right subtrees, which are in turn nL. Finally, the nL construction can be realized by first building a list L of m/n one-node nL, and then iteratively melding the elements of L until only one remains. From easy calculations, it follows that the complexities C_I, C_D and C_{MQ} of these three operations are exactly as in the nH case.

4. Conclusions

The PPQ data type introduced in this paper is based on the idea of extending sequential Priority Queue structures to a parallel context. Adopting the CREW-PRAM model of parallel computation, we have defined two PPQ data structures, the nH and the nL, respectively realizing unmeldable and meldable queues. We have then employed optimal algorithms of sorting and merging devised for the CREW-PRAM for the efficient implementation of the basic operations of insert, deletemin, makequeue and meld.

For a better assessment of the efficiency of our structures, consider a parallel version of Heapsort, made of a Makequeue operation followed by m/n Deletemins, where m and n are the elements to be sorted, and the number of processors, respectively. Letting $C_{HS}(m, n)$ be the time complexity of Heapsort, from (4) and (5), we have:

$$C_{HS}(m, n) \in O\left(\frac{m}{n} \log n + \sum_{i=1}^{\log m/n} \frac{m}{n2^i} \min\left\{\log \frac{m}{n} - i + \log n,\ \left(\log \frac{m}{n} - i\right) \log\log n\right\}\right)$$

that is:

$$C_{HS}(m,n) \in O((m/n \log m))$$

which is clearly optimal for any value of m and n, with $m \geq n$.

The optimality of the above algorithm is due to the use of Cole's O(log n) complex parallel sorting. However, for $m > n \cdot \log\log n$, we can slightly modify the sorting phase of makequeue to use simpler sorting algorithms (as in [7]) without increasing the overall running time.

Finally, we want to point out that the extended heap order, as defined in section 2, does not require that the elements at each node be sorted. We are currently investigating the techniques needed to implement PPQ operations without making use of sorting but still achieving the same time complexities.

Acknowledgement

We wish to thank Fabrizio Luccio and Andrea Pietracaprina for helpful suggestions.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] A. Borodin and J.E. Hopcroft, Routing, Merging and Sorting on Parallel Models of Computation, *J. Comput. Syst. Sci.* **30** (1985) 130-145.

[3] R. Cole, Parallel Merge Sort, *SIAM J. Comput.* **17** (4) (1988) 770-785.

[4] S. Fortune and J. Wyllie; Parallelism in Random Access Machines, *Proc. 10th ACM Symp. on Theory of Computing (1978)* 114-118.

[5] R.M. Karp, M. Saks and A. Wigderson, On a Search Problem Related to Branch-and-Bound procedures, *Proc. 27th Ann. IEEE Symp. on Foundations of Computer Science* (1986) 19-28.

[6] R.E. Tarjan, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1983.

[7] L.G. Valiant, Parallelism in Comparison Problems, *SIAM. J. Comput.* **4** (3) (1975) 348-355.

[8] J.W.J. Williams, Algorithm 232: Heapsort, *Comm. ACM* **7** (6) (1964) 347-348.

An Optimal Parallel Construction Scheme for Heap-like Structures

(Summary)

S. Olariu† and Z. Wen

Computer Science Dept. Old Dominion University, Norfolk, VA 23529-0162

1. Introduction *Heaps* are popular data structures for priority queues. A heap is a *heap-shaped* binary tree with a *max-ordering* [1]. For double-ended priority queues several data structures have been proposed. The *minmax heap* [1] is a binary tree with the heap-shape property and a *min-max ordering*. The min–max heaps are not mergeable in subliniar time [4]; in [5] it is argued that another data structure to implement double-ended priority queues, called min–max–pair heap, can be merged in sublinear time: two min–max–pair heaps with n and k nodes can be merged in time $O(\log (n/k) * \log k)$. A min–max–pair heap is a heap-shaped binary tree H, every node having two fields, a *min field* and a *max field*, with the key in the min (max) field of H[i] being the smallest (largest) in the subtree of H rooted at H[i] ($1 \le i \le n$). Note that H contains 2n or 2n-1 keys. A *deap* [3] has the heap-shape property, with the left (right) subtree of the non-existing root organized as a min-heap (max-heap), and each leaf in the min-heap is smaller than a corresponding leaf in the max-heap.

We present a technique that provides optimal parallel construction algorithms for heaps, min–max heaps, deaps, and min–max–pair heaps. We also show that merging two heaps in parallel can be done efficiently.

2. Constructing heaps in parallel

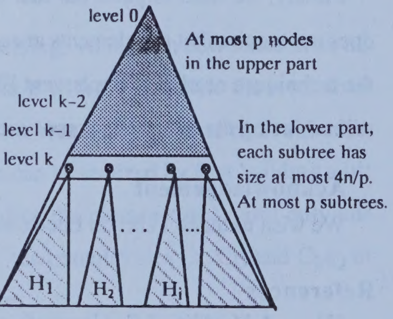
An n-element array H can be interpreted as a heap-shaped binary tree if for every i ($1 \le i \le \lfloor \frac{n-1}{2} \rfloor$), the children of H[i] are H[2i] and H[2i+1]. Perhaps the easiest way to explain our technique is by showing how to induce a heap structure on H. We proceed in the following two stages. (See Figure 1)

Stage 1. With $k = \lfloor \log p \rfloor$, assign one processor to each of the subtrees H_i of H rooted at H[i] with $2^k \le i \le 2^{k+1}-1$. It is easy to confirm that at most p processors are assigned in this way and that each such subtree contains at most 4n/p nodes of H. Therefore, the unique processor assigned to H_i makes H_i into a heap in O(n/p) time.

Stage 2. We propose to "grow" in parallel the heaps H_i ($2^k \le i \le 2^{k+1}-1$) into H itself by adapting the well-known sequential trickledown. The p processors are redistributed to the first 2^k-1 nodes of H, one processor per node. Every element of H at levels k-1 through 0 will be trickled down in parallel. To avoid read and write conflicts we proceed in a pipelined fashion: we begin by activating the processors at level k-1 which will proceed to "trickle down" two levels; after this, in parallel, the processors at level k-2 will begin the trickle down, and so on. Every processor remains active until it reaches a leaf when it becomes inactive. In trickling down from a node w to one of its children, processor P does the following:

```
v←child of w with the largest key;
if key(w) < key(v) then
    swap(key(w),key(v));
Processor P moves down to v;
```

Theorem 1. An n element array H[1..n] can be made into a heap in O(n/p) time using p ($p \le n/\log n$) processors in EREW-PRAM. □



(1) Construct the sub_structures (heaps for the subtrees in the lower part.

(2) Reassign the processors to the upper part, one processor per node. Trickle down the numbers in this part, level by level in a pipelined fashion..

Figure 1. The parallel scheme to construct heap–like structures

3. Constructing min-max heaps in parallel

Consider an n-element array H. The first stage of our parallel min-max heap-construction algorithm is almost the same as Stage 1 in Section 2: we assign one processor to each subtree H_i ($2^k \le i \le 2^{k+1}-1$) of H, and let every assigned proces-

---

† This author's work was supported by the National Science Foundation under grant CCR-8909996.

sor make H_i into a min–max heap or a max-min heap depending on whether $\lfloor \log i \rfloor$ is even or odd. Since every H_i contains at most $4n/p$ keys, this takes $O(n/p)$ time as in [1]. Next, the p processors are redistributed to the first 2^k-1 elements of H. The idea of the second stage is to "grow" in parallel the min–max heaps H_i ($2^k \le i \le 2^{k+1}-1$) into H itself, by adapting the Trickledown procedure in [1]: every element of H at levels k-1 through 0 will be trickled down *two min* (resp. max) levels in parallel in a pipelined fashion.

Theorem 2. An n-element array H[1..n] can be made into a min–max heap in $O(n/p)$ time using p ($p \le n/\log n$) processors in EREW-PRAM.□

4. Constructing min-max-pair heaps

Consider an array H[1..n]. As a first step, with $k=\lfloor \log p \rfloor$, our algorithm assigns one processor to each of the subtrees H_i of H rooted at H[i] with $2^k \le i \le 2^{k+1}-1$. The unique processor assigned to H_i makes H_i into a min–max–pair heap in $O(n/p)$ time, as in [5]. After Step 1 the p processors are redistributed to the first 2^k-1 elements of H. Step 2 has two substages: first, the processors trickle down the min fields at level k-1 through 0 in a pipelined fashion; next, we trickle down the max fields in a similar fashion [6].

Theorem 3. An n-element array H[1..n] can be made into a min–max–pair heap in $O(n/p)$ time using p ($p \le n/\log n$) processors in EREW-PRAM. □

5. Constructing deaps in parallel

Consider an n-element deap H[1..n+1] as in [3]. Note that the entries at level i in the min-heap are $2*2^i, 2*2^i+1, \dots, 3*2^i-1$ provided they exist; similarly, the entries at level i in the max-heap are $3*2^i, 3*2^i+1, \dots, 3*2^i+2^i-1$

To construct a deap we mirror the scheme presented in Section 2; in the first stage, with $k=\lfloor \log p \rfloor$, we assign one processor to each pair of subtrees $(H_t, H_{t'})$ of H rooted at H[t], H[t'] with $t=2*2^k+j$ and $t'=3*2^k+j$ ($1 \le j \le 2^k-1$). Every assigned processor makes its pair of subtrees into a deap in $O(n/p)$ time as in [3]. After this, the p processors are redistributed to the first 2^k-1 elements of H. It is helpful to imagine a deap as in Figure 2. The nodes of H of the form H[s] with $s=2*2^i+j$ such that $0 \le i < k$ and $1 \le j \le 2^i-1$ will be called *upper nodes* and all the nodes of the form H[s'] with $s'=3*2^i+j$ such that $0 \le i-1$ and $1 \le j \le 2^i-1$ will be called *lower nodes*. In stage 2, the processors are assigned to upper nodes only. Every upper node is trickled down (this trickle down differs from the standard one, see [6]) in parallel in a pipelined fashion; In stage 3, the lower nodes receive processors and they will trickledown (moving up in Figure 2) as in Stage 2.

Theorem 4. An n-element array H[1..n+1] with H[1] unused can be made into a deap in $O(n/p)$ time using p ($p \le n/\log n$) processors in EREW-PRAM. □

5. Merging heaps in parallel

In [6] we show that using a pipeline technique, the sequential heap-merging algorithm in [7] can be easily parallelized. We have the folowing result.

Theorem 5. Two heaps with n and k nodes, respectively, can be merged in time $O(\log n)$, using $\min\{\lfloor \log n \rfloor - \lfloor \log k \rfloor, \lfloor \log k \rfloor\}+1$ processors in EREW-PRAM.

References

1. M. D. Atkinson, J. R. Sack, N. Santoro, and T. Strothotte, Min-max heaps and generalized priority queues, *CACM* 29 (1986), 996-1000.

2. S. Baase, Computer Algorithms - An Introduction to Design and Analysis, Addison-Wesley, 1988.

3. S. Carlsson, The deap - A double ended heap to implement double-ended priority queues, *IPL* 26, (1987) 33-36.

4. A. Hasham and J. R. Sack, Bounds for min-max heaps *BIT* 27 (1987) 315-323.

5. S. Olariu and Z. Wen, The min–max–pair heap and its variations, Dept of Comp. Science ODU, TR-89-33.

6. S. Olariu and Z. Wen, Fast Parallel Heap Algorithms, Dept of Comp. Science ODU, TR-90-12.

7. J-R Sack and T. Strothotte, An algorithm for merging heaps, *Acta Informatica* 22 (1985) 171-186.

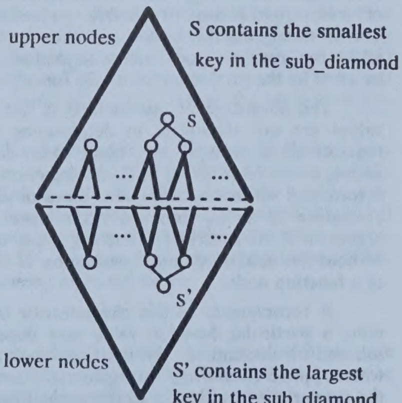

upper nodes
S contains the smallest key in the sub_diamond
S
...
...
...
...
S'
lower nodes
S' contains the largest key in the sub_diamond

Figure 2. A new way of looking at the deap

Parallel Demand – Driven Evaluation Of Logic Networks

P. E. Dunne, C. J. J. Gittings and P. H. Leng

Department of Computer Science
University of Liverpool
Liverpool L69 3BX
England

Abstract

This paper examines strategies for the demand-driven evaluation of logic networks in a multiprocessor environment. Analysis of the two-processor simulation problem demonstrates how much more difficult it is to construct optimal multiprocessor algorithms than in the single processor case. An heuristic for two-processor simulation is proposed and investigated, by analysis and simulation.

1. Introduction

A *logic network* is a directed acyclic graph, the arcs of which represent binary logic variables, and the nodes of which correspond to arbitrary functions of these variables. Its closeness to the physical structure of an (electronic) logic circuit makes this a convenient representation in systems for the computer-aided design of digital systems and VLSI circuits. A logic network may also be used as an abstraction of database query evaluation, in which nodes represent facts, and arcs represent the hierarchical relationships between facts, or of logic programming, in which goals are represented by nodes, linked by arcs to sub-goals. Strategies for the efficient evaluation of logic networks are therefore of considerable interest, with applications to VLSI circuit simulation, the execution of logic programs and database query evaluation.

The usual method applied in logic circuit simulation employs a discrete *event-driven* strategy to schedule evaluation of simulated circuit components (nodes of the graph) [1,2]. This strategy, which corresponds to a data-driven evaluation rule, schedules evaluation of a node function in response to every change in value of an arc which is an input to the node. Consequent changes to values of node output arcs stimulate further evaluations throughout the network, the process continuing until terminal nodes are evaluated or a stable state is reached.

A drawback of this strategy is that it can lead to unnecessary evaluation of node functions which do not contribute to resolution of the final circuit state, and also to repeated evaluation of node functions in response to sequences of input changes. In order to improve the evaluation speed, we have investigated the use of a *demand-driven* strategy for network evaluation [3,4]. In this method (also employed for logic circuit simulation in the BACKSIM system [5]), a demand for the value of a network output is used to schedule evaluation of the function node which is the source of this output. This in turn generates further demands for the evaluation of arcs which are inputs to this node, that is, for the evaluation of function arguments. The recursive demand sequence thus initiated is finally satisfied by the production of values for network input arcs.

The advantage of this method is that it avoids the necessity of evaluating sub-graphs whose values are not significant in determining the overall network output. A function which always requires all of its input arc values to be defined before its output arc value can be determined is known as a *strict* function [6,7]. Thus, a non-strict function is one whose value may, in some cases, be determined without knowing the values of all its arguments, and hence, without the need to demand evaluation of all the function node input arcs. A specific example is a node representing a 2-argument AND function; if one argument is evaluated FALSE, the result of the function is known, without the need to demand evaluation of the other argument. This is known as the *lazy evaluation* of a function node.

A consequence of this characteristic is that the number of evaluation steps required to determine a particular function value may depend on the order in which the function arguments are selected for evaluation. We therefore seek a strategy for making this selection at each node, which, when applied recursively throughout the network, will minimise the total expected time to evaluate the network. We will refer to this evaluation time as the *cost* of evaluation.

Consider an AND function with n arguments, represented by arcs labelled $x_1, x_2, ..., x_n$. Two factors are significant in choosing the order in which demands for arc values are made :-

(i) the expected cost w_i of evaluating the arc x_i, representing a weighted sum of the expected number of arc evaluations required in the recursive elaboration of the demand for x_i;

(ii) the probability p_i that x_i will be evaluated TRUE.

It has been shown independently in [8] and [9] that, assuming independent probabilities for node inputs, the optimal strategy is to select arcs for evaluation in ascending order of the ratio $w_i/(1-p_i)$ (for an AND function) or w_i/p_i (for an OR function). In [8], this result has been extended to deal with the case of threshold functions. An n-argument threshold function $T_k^n(X_n)$ has the value TRUE if and only if at least k of its arguments have the value TRUE. In this case, we define a set A, in which arcs are placed in ascending order of $w_i/(1-p_i)$, and a set O in which arcs are placed in ascending order of w_i/p_i. The optimal strategy chooses an arc for evaluation which is included in both the first $n-k+1$ elements of A and the first k elements of O.

The assumption underlying these strategies is that the total network evaluation time is proportional to the total number of evaluation steps required. If, however, we propose an evaluation environment in which more than one evaluation step can proceed concurrently, then this assumption is invalidated and the strategies derived no longer apply. In this paper we examine strategies for the evaluation of logic networks in a multiprocessor environment.

2. Multiprocessor Theoretical Model

2.1. Initial Analysis

In extending the single processor concepts to a multiprocessor model, some new notation is introduced. Let w_i^j be the expected time to evaluate arc x_i using exactly j processors. The evaluation of x_i implies, of course, evaluation of arcs which precede x_i in the graph. When more than one processor is available, it is assumed that some of these arc-evaluations will proceed in parallel as a result of some policy for allocating processors to evaluations. We define the *m-cost* of an arc x_i as the expected time to evaluate x_i using exactly m processors, with the chosen allocation strategy, i.e. $w_i^j \equiv$ *j-cost* of x_i

Our aim will be to choose a strategy which minimises these costs. Consider initially the two-processor case: in the one-processor case, when attaching the processor to an input arc of a node, the only decision is to determine the 'best' next input arc to consider; with two processors available, however, there are two possibilities :-

a) Attach one processor to evaluate a chosen input arc x_{i_1} and the other to evaluate some other arc x_{i_2} in parallel;

b) Attach both processors to evaluate a single chosen arc x_i.

To simplify the algebraic analysis, we assume that when a number of processors are allocated simultaneously to different input arcs of a node, these processors can be re-allocated only when they have all evaluated their respective arcs; no pre-emption is possible. The only exception to this is when the values so far returned by processors are sufficient to determine the node's output, even if they have not all returned arc values. For example, with an OR function, as soon as a processor evaluates an input arc to be TRUE, any processors evaluating other input arcs of the same node can be pre-empted as their values are not needed.

In the two processor case, consider a 3-input AND function, where the two processors have been allocated to evaluate two of the three input arcs in parallel. If the first processor returns FALSE, then the output of the node is known to be FALSE and the other processor can be pre-empted. If, however, the first evaluated arc is TRUE, then the processor which returned this is left idle until the second processor produces its result. Should the value returned by the second processor be FALSE, then the node output is known to be FALSE. If the second processor returns TRUE, then both processors are available to determine the remaining input arc value.

2.2. Two-Processor Analysis

We begin by considering networks containing 2-input AND functions. Associated with each arc are a 1-cost, a 2-cost and a probability of evaluating to TRUE (Figure 2.1). We wish to determine the optimal 1-cost and 2-cost of the output arc x_3 (w_3^1 and w_3^2 respectively), together with p_3.

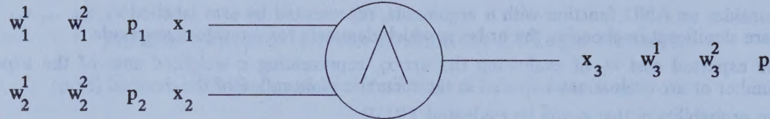

w_1^1 w_1^2 p_1 x_1
w_2^1 w_2^2 p_2 x_2
$\wedge$
x_3 w_3^1 w_3^2 p_3

Figure 2.1 – Two-input AND function

The values of w_3^1 and p_3 are as determined in the single processor case :-

$$p_3 = p_1 p_2$$

$$w_3^1 = \begin{cases} w_1^1 + p_1 w_2^1 & \text{if } \dfrac{w_1^1}{1-p_1} \le \dfrac{w_2^1}{1-p_2} \\ w_2^1 + p_2 w_1^1 & \text{otherwise} \end{cases}$$

The former corresponds to allocating the processor to x_1 first, the latter to evaluating x_2 first. In each case, the formula represents the time to evaluate the first chosen arc, plus the time to evaluate the second arc multiplied by the probability that this will be necessary.

With two processors, there are three possible allocation strategies to determine x_3 :-

S1) Attach both processors to x_1 first (then to x_2 if x_1 is TRUE);
S2) Attach both processors to x_2 first (then to x_1 if x_2 is TRUE);
S3) Attach one processor to each of x_1 and x_2, to evaluate x_1 and x_2 in parallel.

Each of these strategies has an expected cost; the optimal strategy will be that with the lowest expected cost. For cases (S1) and (S2), similar expressions apply to those derived for the one-processor case. Thus:

The expected cost of (S1) is $w_1^2 + p_1 w_2^2$
The expected cost of (S2) is $w_2^2 + p_2 w_1^2$

For case (S3), there are two possible expressions for the expected cost :-

S3a) $w_1^1 + p_1(w_2^1 - w_1^1)$ if $w_1^1 \le w_2^1$
S3b) $w_2^1 + p_2(w_1^1 - w_2^1)$ otherwise

For example, in (S3b), w_2^1 is the expected cost of using the first processor to evaluate x_2, and $(w_1^1 - w_2^1)$ represents the extra time required by the second processor in evaluating x_1; there is probability p_2 that this value will be needed (i.e. it is needed when x_2 is TRUE).

In choosing the lower of (S1) and (S2) strategies, i.e. whether the expected cost of allocating both processors to x_1 first is lower than that of allocating both to x_2 first, a similar condition applies to that for the one-processor case, but with '2' superscripts on each weight value :-

Cost of (S1) ≤ cost of (S2) iff :-

$$w_1^2 + p_1 w_2^2 \le w_2^2 + p_2 w_1^2$$

$$w_1^2(1-p_2) \le w_2^2(1-p_1)$$

$$\frac{w_1^2}{1-p_1} \le \frac{w_2^2}{1-p_2}$$

When the lower of costs (S1) and (S2) has been determined, this can be compared with the cost of splitting the two processors, i.e. either (S3a) or (S3b), to find the lowest overall expected cost, and hence the optimal evaluation strategy.

The above analysis applies only to two-input AND functions. We note, however, that an n-input AND function, whose input arcs have individual 1-cost, 2-cost and probability values, is functionally equivalent to a cascade of $n-1$ two-input AND functions, such that the expected output 1-cost and probability remain the same. The n input arcs are assigned to two-input nodes on the basis of the ordering of $w_i^1/(1-p_i)$ for each arc x_i - the lowest is an input to the node containing the network output arc, while the highest two form inputs to the node at the top of the cascade.

We observe also that, using De Morgan's Laws, replacing a two-input OR function by a two-input AND function with zero-cost inversion of each arc logic value, results in identical 1-cost, 2-cost and probability for the output arc. This allows us to extend our analysis to networks containing a mixture of AND and OR functions.

Note that in a network which is a complete binary tree of alternating levels of two-input AND and OR functions, it is always possible to get speedup by the addition of a second processor, given a 'perfect' allocation strategy - i.e. a strategy which never evaluates an unnecessary arc. For example, for a two-input AND function, provided that at least one of its input arcs is FALSE, the 'perfect' strategy will choose this in preference to an input which turns out to be TRUE.

2.3. Multiprocessor Analysis

To begin the extension of these ideas into the multiprocessor case, we will examine the optimal evaluation of n-input AND functions. Let $\mathbf{X_n} = < x_1, \ldots, x_n >$ be a set of n Boolean variables. With each x_i there are associated *costs* w_i^j which denote the expected time to evaluate the arc x_i using j (identical) processors; in addition each x_i is assumed to have a probability p_i of being 1. In this section we make the following assumptions :-

A1) $p_i = p \quad \forall\, 1 \le i \le n$

A2) Exactly k processors are available

A3) The function to be evaluated is $\bigwedge_{i=1}^{n} x_i$

A4) $n = km$ for some $m \ge 1$

Definition 2.1: An *r-stage (k-processor) evaluation strategy,* E^r, is an ordered partition of $\{1, \ldots, n\}$ into r *blocks;* thus

$$E^r \;=\; < B_1 ; B_2 ; \cdots ; B_r > \qquad \text{where each block } B_i \text{ satisfies } 1 \le \mid B_i \mid \le k.$$

The *expected cost* of E^r, $W(E^r)$, is given by

$$W(E^r) \;=\; \sum_{i=1}^{r} \left[\prod_{j=1}^{i-1} p^{\,|B_j|} \right] cost\,(B_i)$$

where $cost(B_i)$ is the expected (parallel) time taken to evaluate $\bigwedge_{j \in B_i} x_j$ using exactly k processors with every arc in B_i being allocated at least one processor.

E^r is an *optimal r-stage evaluation strategy* if and only if $W(E^r)$ is minimal among all r-stage evaluation strategies. •

We are concerned with optimal m-stage evaluation strategies. Recalling that $n = km$ it follows that for these strategies every block, B_i, contains exactly k elements of $\{1, \ldots, n\}$ and thus no input arc x_i is ever allocated more than one of the available processors in order for its result to be determined. Given this fact we may simplify our notation by referring to $w_i \equiv w_i^1$ as *the* cost of x_i. Throughout the following it is assumed, without loss of generality, that $w_i \le w_{i+1} \quad \forall\, 1 \le i < n$

It should be noted that for such strategies $cost(i_1, \ldots, i_k)$ is

$$= w_{i_1} + \sum_{j=2}^{k} p^{j-1}(w_{i_j} - w_{i_{j-1}}) \;=\; \sum_{j=1}^{k-1} (p^{j-1} - p^j)\, w_{i_j} + p^{k-1} w_{i_k} \qquad (2.1)$$

(where $w_{i_j} \le w_{i_{j+1}}$ for each $1 \le j < k$). Note that the expression in (S3a) is expression (2.1) in the special case $k = 2$. The main result of this section may be stated as

Theorem 2.1: Let $n = km$ and $\mathbf{X_n}$, $< w_1, \ldots, w_n >$ be as defined previously. For all p an optimal m-stage evaluation strategy is given by the ordered partition $P = < B_1 ; B_2 ; \cdots ; B_m >$ in which

$$B_i = (\,(i-1)k + 1,\; (i-1)k + 2, \ldots,\; ik\,) \text{ for } each\; 1 \le i \le m$$

Proof (outline): We first show that the theorem holds for $n = 2k$. Let

$$P = < (i_1, i_2, \ldots, i_k); (i_{k+1}, i_{k+2}, \ldots, i_{2k}) >$$

(where $i_j < i_{j+1}$ for $1 \le j < k$ and $k+1 \le j < 2k$) be an optimal 2-stage strategy for $\mathbf{X}_{2k}$.

Let $S(d)$ be the proposition: 'There is an optimal 2-stage strategy such that the first d arcs in the first partition block are $1, 2, \ldots, d$'.

It can be shown by induction on d $(1 \le d \le k)$ that $S(d)$ holds.

To complete the proof for $n = km$ $(m > 2)$ it is sufficient to observe that if P is a strategy other than that described in the theorem statement then there must be two consecutive blocks $B_i ; B_{i+1}$ in which the k'th arc in B_i is greater than the first arc of B_{i+1}. Comparing P with the strategy Q obtained from it rearranging the arc indices forming $B_i ; B_{i+1}$ into ascending order shows that Q has cost no greater than that of P. □

We can also easily determine optimal strategies for evaluating complete binary trees of two-input AND functions using two processors.

We assume that the total number of network arcs, n, is an exact power of 2, $n = 2^k$ say. Let S be a complete binary tree of (2-input) $\wedge$-functions with n input arcs $\mathbf{X_n} = < x_1, \ldots, x_n >$. We make the following additional assumptions: each input arc x_i has $w_i^1 = w_i^2 = 1$ and $p_i = p$.

There are two possible ways to proceed to evaluate the output of the network: either assign both processors to one input of this or assign one processor to each of the inputs. Now if n is large enough then the first policy will be best; if n is small the second method will be better. The threshold point will depend on p.

To determine a complete strategy it is necessary to find a condition for evaluating this threshold. Let $C_j(n)$ denote the cost of evaluating the output of S using j processors. It can be shown that :-

$$C_1(n) = 1 + \sum_{j=0}^{k-1} \frac{1}{1 - \beta_j^p}$$

where $\beta_r^p = p^{2^r}$, and that :-

$$C_2(n) = 1 + \frac{2(1-\beta_k^p)}{1-\beta_{t_p}^p} + \frac{1-\beta_k^p}{\alpha_{t_p-1}^p} \left(\sum_{j=0}^{k-1} \frac{1}{1-\beta_j^p} + \beta_{t_p-1}^p \sum_{j=t_p+1}^{k-1} \frac{1}{1-\beta_j^p} - \frac{2+\beta_{t_p-1}^p}{1-\beta_{t_p}^p} \right)$$

where $\alpha_r^p = 1 + \beta_r^p$ and t_p is the maximal size of tree at which one would immediately split the two processors to evaluate the output of the tree.

Even in the restricted forms which we have examined, it is clear that the multiprocessor analysis is considerably more complicated than the single processor case. In the latter, for the case of an n-input AND function, it has been shown [8] that the optimal input arc to evaluate next can be determined following an ordering of the values of $w_i^1/(1-p_i)$ for all arcs x_i, and that this can be performed in $O(n \log n)$ steps, which makes it of practical use.

In the general m-processor case, however, where each arc x_i has weights w_i^1, ..., w_i^m (the expected cost of evaluating x_i using between 1 and m processors), no strategy has been derived which avoids consideration of all these weights in determining the optimal expected cost of the output arc of a node. To avoid the computation of all these expected costs, we seek a strategy to enable a near-optimal processor allocation without the need to determine all possible strategy-costs. To assist in an examination of possible heuristics, a study of the problem by simulation has been carried out, described in the following section.

3. Experimental Simulation

3.1. Description of the Model

To facilitate investigation of evaluation heuristics, a generalised abstract model of parallel evaluation was defined and implemented as a software simulation. In this model, a network is represented as a linked list of node data structures: each node data structure holds details about an individual node. This includes such information as the numbers of the node's inputs and output arcs, the output value of the node (or 'x' if not yet calculated) and the number of processors tied to evaluating the node.

The data structure also holds a 'lazy' truth table representing the node's functional behaviour, e.g. the table for a four-input AND function, with inputs x_1 to x_4 and output x_5 is as follows :-

x_1	x_2	x_3	x_4	x_5
0	x	x	x	0
x	0	x	x	0
x	x	0	x	0
x	x	x	0	0
1	1	1	1	1

The 'x's represent unknown/unnecessary values; for example, the third row indicates that if x_3 is FALSE, then x_5 will be FALSE, regardless of the values of x_1, x_2 and x_4.

A simulation proceeds in a demand-driven manner: initially, all processors are allocated to input arcs of the node which has the ultimate network output as its output arc. Allocation is carried out using a chosen strategy, and propagates recursively backwards through the network until ultimately processors are allocated to evaluate network input values, allowing the recursion to unwind and node output arc values to be calculated.

3.2. The Heuristic Under Investigation

With two processors available, each arc x_i has costs w_i^1 and w_i^2 associated with it, being the expected costs of evaluating the arc using one and two processors, respectively.

We wish to investigate the possibility that the expected 2-cost of an arc (w_i^2) can be represented as some function of the expected 1-cost (w_i^1). Were this to be the case, the two-processor system could be considered essentially as an extension of the single processor case, where only one expected cost value is associated with each arc. We examine this possibility by studying the relationship between 1-costs and 2-costs resulting from the application of a 'perfect' allocation strategy.

For each two-input node, the minimum w_i^1 and w_i^2 values obtainable by a 'perfect' allocation strategy are calculated for each of the four possible input arc assignments (i.e. 00, 01, 10, 11 binary). Each node is either a two-input AND, OR or XOR function. The table for the two-input AND function is as follows :-

Input arcs		Output arc	Minimum expected 1-cost	Minimum expected 2-cost
x_1	x_2	x_3	w_3^1	w_3^2
0	0	0	$\min(w_1^1, w_2^1)$	$\min(w_1^2, w_2^2)$
0	1	0	w_1^1	w_1^2
1	0	0	w_2^1	w_2^2
1	1	1	$w_1^1 + w_2^1$	$\min(\max(w_1^1, w_2^1), w_1^2 + w_2^2)$

This illustrates the 'perfect' allocation strategy in action. In the '0 0' input case, the 'perfect' strategy allocates one or two processors to evaluate the input arc with lower 1-cost and 2-cost. In the '0 1' and '1 0' cases, the strategy allocates one or two processors to evaluate the input arc which is '0', thus avoiding the cost of evaluating the '1' input arc.

In the '1 1' case, for an AND function, both input arcs must be evaluated. In the one-processor case, the input arcs are evaluated serially, resulting in the expected output 1-cost being the sum of the input 1-costs. With two processors, there are two possible evaluation strategies, the lower of which is chosen: either, each of the input arcs is evaluated in parallel by one processor (in which case, the expected output cost is the larger of the two input arc 1-costs, since they must both return values) or, the input arcs are evaluated serially by both processors; the expected output cost is the sum of the 2-costs of the input arcs in this case.

The table for the two-input OR function is the dual of the two-input AND table, with all input and output arc value entries negated. The two-input XOR function is more straightforward: this function is strict in both its arguments, so all table entries are the same as the '1 1' AND case.

3.3. Description of a Network

A network to be simulated can be described in one of two ways :-

a) Manually: for each two-input node, the function type (AND/OR/XOR), the numbers of the input and output arcs and the initial 1-costs and 2-costs of a node's input arcs (0 if not network input arcs) are entered explicitly. This allows particular test networks to be described, to ensure that the correct expected 1-costs and 2-costs of output arcs are being calculated.

This method is only practical for networks containing relatively few nodes because of the amount of information which must be provided to describe the network.

b) Alternatively, the network can be constructed with minimal human input: the only parameter is the total number of arcs to be present in the whole network; for a network containing a arcs, (where a is odd) there will be $(a-1)/2$ nodes with two inputs. A network containing this number of arcs and nodes is then constructed.

If a is the total number of arcs, then a 'random' odd number between 1 and a is determined - call this b. Then, b arcs are assumed to be connected to the upper arc of the output node, with $a-b-1$ connected to the lower input (one arc is the output of the node). This is then performed recursively on the two sub-networks formed, containing b arcs and $a-b-1$ arcs, respectively, until the total network is defined.

The type of each node is also chosen at 'random', from among the AND, OR and XOR functions. In the 'random' networks, input arcs are all given equal expected 1-cost and 2-cost values.

This 'random' network configuration allows more realistically-sized networks, with 100s/1000s of nodes, to be examined.

The expected 1-cost and 2-cost of every arc using the 'perfect' allocation strategy are then calculated in a demand-driven, recursive manner. A demand is made to calculate the expected 1-cost and 2-cost of the node containing the final network output arc. This leads to independent demands to calculate the same for the upper and lower input arcs of this node. Ultimately, demands propagate to nodes containing network input arcs; at this point, the values of the 1-cost and 2-cost of the output arcs can be calculated (using the table of the respective node function), and the recursion can unwind. The costs of these output arcs allow further costs to be calculated until the costs for every network arc are known.

When the expected 1-cost and 2-cost of every arc have been calculated, for each arc x_i, the ratio $one-cost/two-cost$ (w_i^1/w_i^2) is calculated.

This value is then averaged for the two cases :-

Av1) non-input arcs only;

Av2) both input and non-input arcs.

In the former case, the ratios are totalled for all non-input arcs and then divided by $(a-1)/2$, i.e. the the number of non-input arcs. In the latter case, all the ratios are totalled and this value is divided by a.

The reason for finding two averages was to observe the effect of the network input arcs on the ratio averages, since input arcs all have a ratio of 1 - the expected 1-costs and 2-costs of input arcs are equal. This second average would be expected to be lower than the average in case (Av1).

3.4. Results

Test network sizes containing between 50 and 1000 nodes (i.e. between 101 and 2001 arcs) were used. For each selected network size, 10 different configurations were constructed using the 'random' configuration described above. The values of the two ratio averages (Av1) and (Av2) were recorded, and then averaged over the 10 configurations, with the mean and standard deviation as follows :-

	Total Number Of Nodes									
	50		100		250		500		1000	
	Av1	Av2	Av1	Av2	Av1	Av2	Av1	Av2	Av1	Av2
$\bar{x}$	1.5308	1.2628	1.5298	1.2636	1.5223	1.2620	1.5279	1.2637	1.5245	1.2621
σ_n	0.0420	0.0208	0.0157	0.0078	0.0249	0.0119	0.0110	0.0055	0.0107	0.0054

Table 3.1 - Networks containing AND/OR/XOR functions

The results from networks consisting of AND and OR functions only, and then XOR functions only, were also taken :-

	Total Number Of Nodes									
	50		100		250		500		1000	
	Av1	Av2	Av1	Av2	Av1	Av2	Av1	Av2	Av1	Av2
$\bar{x}$	1.3526	1.1746	1.3582	1.1782	1.3613	1.1803	1.3613	1.1805	1.3615	1.1806
σ_n	0.0071	0.0035	0.0054	0.0027	0.0031	0.0015	0.0021	0.0010	0.0013	0.0006

Table 3.2 - Networks containing AND/OR functions only

	Total Number Of Nodes									
	50		100		250		500		1000	
	Av1	Av2	Av1	Av2	Av1	Av2	Av1	Av2	Av1	Av2
$\bar{x}$	1.7818	1.3722	1.7593	1.3778	1.7572	1.3779	1.7609	1.3801	1.7548	1.3772
σ_n	0.0509	0.0252	0.0435	0.0216	0.0214	0.0107	0.0073	0.0037	0.0120	0.0060

Table 3.3 - Networks containing XOR functions only

4. Conclusions

4.1. Comments On The Results

In each of the three sets of empirical observations the values of the $\bar{x}$'s are nearly constant, and independent of the network size; the low values of the σ_n's would seem to indicate that these $\bar{x}$ values are independent of the network configuration.

As expected, the value of ratio average (Av1) is greater than ratio average (Av2), demonstrating the effect of the input arcs, with equal 1-costs and 2-costs, on the ratio average (Av2).

The effect of XOR functions on the average ratios is clearly seen (in the 'perfect' allocation strategy, the XOR function benefits most from the second processor, since this is strict in both its arguments: there is not the opportunity for a single processor to select an input arc which may make lazy evaluation possible). The average ratios, both (Av1) and (Av2), are proportional to the number of XOR functions in the configuration, from the lowest - where no XOR functions are present (Table 3.2), to the highest - where all nodes are XOR functions (Table 3.3). The above results suggest that there is a linear relationship between the expected 1-cost and 2-cost of an arc for the 'perfect' allocation strategy, i.e.

$$w_i^2 \simeq \varepsilon\, w_i^1 \qquad \varepsilon \in \mathbf{R}^+$$

We know that, at best, ε could be $\frac{1}{2}$, i.e. there is a speedup factor of 2 by introducing a second processor and at worst, ε will be 1.

4.2. An Heuristic

From the empirical results for (Av1), for networks containing AND, OR and XOR functions, we have a value for $\frac{1}{\varepsilon}$ of $\simeq 1.53$, giving a value for ε of $\simeq \frac{2}{3}$.

Using $\varepsilon = \frac{2}{3}$, we propose the following heuristic for two-processor evaluation :-

1) Calculate the 1-cost of every arc in the network, using the one-processor algorithm in [8];
2) At the same time, multiply each 1-cost by $\frac{2}{3}$, to form the approximated 2-cost value;
3) Without loss of generality, we assume $w_1^1 \leq w_2^1$. The costs of possible two-processor strategies are then (arcs refer to Figure 2.1) :-

Ca) $\frac{2}{3}w_1^1 + \frac{2}{3}p_1 w_2^1$ (both to x_1 first)

Cb) $\frac{2}{3}w_2^1 + \frac{2}{3}p_2 w_1^1$ (both to x_2 first)

Cc) $w_1^1 + p_1(w_2^1 - w_1^1)$ (split one processor to x_1 and one processor to x_2)

We want to select the strategy with the lowest cost. Now, cost (Ca) $\leq$ cost (Cb) if :-

$$\frac{2}{3}w_1^1 + \frac{2}{3}p_1 w_2^1 \leq \frac{2}{3}w_2^1 + \frac{2}{3}p_2 w_1^1$$

$$w_1^1 + p_1 w_2^1 \leq w_2^1 + p_2 w_1^1$$

$$\frac{w_1^1}{1-p_1} \leq \frac{w_2^1}{1-p_2}$$

This is the same condition as in the one processor case. Once the better strategy of (Ca) and (Cb) has been selected, this can be compared with the cost of strategy (Cc).

If cost (Ca) $\leq$ (Cb), then we compare strategies (Ca) and (Cc): (Ca) $\leq$ (Cc) if :-

$$\frac{2}{3}w_1^1 + \frac{2}{3}p_1 w_2^1 \leq w_1^1 + p_1(w_2^1 - w_1^1)$$

$$w_1^1(p_1 - \frac{1}{3}) \leq \frac{1}{3}p_1 w_2^1$$

$$w_1^1(3 - \frac{1}{p_1}) \leq w_2^1$$

If this condition holds, then strategy (Ca) should be used, else (Cc) should be used. Since we assume $w_1^1 \leq w_2^1$, this condition need only be tested for :-

$$3 - \frac{1}{p_1} > 1 \quad \text{i.e. } p_1 > \frac{1}{2}$$

Thus, for $p_1 \leq \frac{1}{2}$, strategy (Ca) is always better than strategy (Cc).

If, however, (Cb) < (Ca), then we compare (Cb) and (Cc): (Cb) $\leq$ (Cc) if :-

$$\frac{2}{3}w_2^1 + \frac{2}{3}p_2 w_1^1 \leq w_1^1 + p_1(w_2^1 - w_1^1)$$

$$w_1^1(\frac{2}{3}p_2 - 1 + p_1) \leq w_2^1(p_1 - \frac{2}{3})$$

Since we know nothing about p_1, other than $0 \leq p_1 \leq 1$, we cannot divide by $(p_1 - 2/3)$ since the inequality may reverse. In this case, therefore, it is necessary to calculate the costs of strategies (Cb) and (Cc), and choose the smaller.

However, the comparison of strategies (Cb) and (Cc) is less likely to occur than the comparison of strategies (Ca) and (Cc), since for the former to occur we must have :-

$$w_1^1 \leq w_2^1 \text{ by the assumption, but } \frac{w_1^1}{1-p_1} > \frac{w_2^1}{1-p_2}$$

4.3. Summary

Strategies for the demand-driven evaluation of logic networks in a multiprocessor environment have been examined. Extensive analysis in extending an existing one-processor algorithm to the two-processor case has been performed. This has clearly demonstrated that even in the simplest multiprocessor case, the analysis becomes considerably more complex than the single processor case, and that there appears to be no practical algorithm for optimal processor allocation in the two-processor case. We have shown, however, that in a number of cases, it is possible to find efficiently computable optimal strategies when certain restrictions are imposed on the allocation policy.

This led to the investigation of heuristic allocation strategies and thence to the implementation of an abstract model of parallel evaluation to allow their evaluation. A two-processor heuristic has been proposed and examined, both by analysis and simulation. The effectiveness of this heuristic, in comparison with an optimal allocation strategy, requires further examination. It is also intended to examine the extension of such heuristics into the general m-processor case.

5. References

[1] Ulrich, E.G: "Exclusive Simulation Of Activity In Digital Networks", Comm. ACM, Vol. 12, No. 2, Feb. 1969, pp. 102-110

[2] Szygenda, S.A; Thompson, E.W: "Digital Logic Simulation In A Time-Based, Table-Driven Environment, Part 1, Design Verification", IEEE Computer, March 1975, pp. 24-36

[3] Jackson, D: "The Generation Of Systems For Microarchitectural Simulation", Ph.D. Dissertation, The University Of Liverpool, 1986

[4] Charlton, C.C; Jackson, D; Leng, P.H: "Modelling And Simulation Of Digital Logic: An Alternative Approach", Proceedings of the 16th. IASTED International Symposium - 'Identification, Modelling and Simulation', Paris, June 1987

[5] Smith, S.P; Mercer, M.R; Brock, B: "Demand Driven Simulation: BACKSIM", Proceedings of the 24th. ACM/IEEE Design Automation Conference, June 1987, pp. 181-187

[6] Burn, G.L; Hankin, C.L; Abramsky, S: "The Theory And Practice Of Strictness Analysis For Higher-Order Functions", Imperial College Department Of Computing Report DOC 85/6, April 1985

[7] Clack, C; Peyton Jones, S: "Strictness Analysis - A Practical Approach", Lecture Notes In Computer Science, 201, Functional Programming Languages and Computer Architecture, Jouannaud J.-P. (ed), Springer-Verlag, Aug. 1985, pp. 35-49

[8] Dunne, P.E; Leng, P.H: "On Algorithms For Improving The Efficiency Of Digital Simulation Methods", Report 88/1, University of Liverpool, Department of Computer Science; Abstract in Bulletin of the EATCS, 1988, No. 36, p. 236

[9] Sassa, M; Nakata, I: "Time-optimal Short-circuit Evaluation Of Boolean Expressions", Inf. Proc. Letters, 29, 1988, pp. 43-51

THE COSTTABLE PROBLEM IS NP-COMPLETE*

Kriss A. Schueller
Dept. of Math. and Comp. Sci.
Youngstown State University
Youngstown, OH 44555

Jon T. Butler
Dept. of Electr. and Comp. Eng.
Naval Postgraduate School
Monterey, CA 93943-5004

ABSTRACT

In the costtable approach to logic design, a function is realized as a combination of functions from a table. The objective is to find the least cost realization, where realization cost is the sum of the costs of the functions used, plus the cost of combining them. If cost represents integrated circuit area [6], the design produces a minimum space realization. We analyze the computational complexity of the costtable approach to the design of multiple-valued logic circuits. An important parameter in the design of circuits by costtable is the *size* of the costtable. Larger costtables provide more efficient realizations. However, experience indicates a substantial increase in design time with increasing costtable size. We show that this design approach is NP-complete.

I. INTRODUCTION

It has been approximately 20 years since the first characterization of the complexity of a logic design problem - minimization of sum-of-products expressions. One can find the minimal sum-of-products expression for a given logic function by solving a minimum set covering problem. Since the minimal sum-of-products extraction appears to be a special case of the minimum set covering problem, there is the prospect that the former is less complex than the latter. However, in 1965, Gimpel[4] showed that this is not the case. Any instance of the set covering problem occurs as an instance of the sum-of-products extraction problem. In 1972, Karp [5] proved that the set covering problem is NP-complete; thus, proving that extracting a sum-of-products expression is NP-complete.

The need for design techniques for newly developed multiple-valued logic circuits [6-7] has inspired interest in the costtable approach [1-2,6-8]. In the costtable approach, a given function is realized by selecting functions from a table and combining them. Associated with each chosen function is a cost, which can represent chip area, power dissipation, speed, etc. The table consists of functions that are typically useful in the realization of many functions and have low cost. The cost of a realization is the sum of the costs of the component functions plus the cost of combining them. Usually, there is more than one way to realize a given function, and the goal of the design is to find a realization of lowest cost. This is called the *Costtable Realization* problem.

Our interest in the costtable technique is inspired by its fundamental nature. Minimization of binary sum-of-products expressions, for example, is a costtable technique where entries are AND functions and the combining operations is OR. VLSI test is a costtable technique where costtable entries are test vectors.

---

*Research supported by NATO Grant 423/84, by NSF Grant MIP-8706553, and an NPS Block Funded Grant for the Naval Research Laboratory.

Except for specific cases, the complexity of the costtable technique has not been analyzed. We consider in this paper the following problem:

> Given a multiple-valued function f and a costtable F of size t, find a minimal cost realization of f consisting of the sum of functions from F.

Given fixed function parameters, as determined by an assignment of values to n, the number of variables and r, the number of logic values, we view the *size* of an instance of this problem as t, the size of the costtable. The question is then how the time to find a minimal cost realization of a given function depends on t.

II. BACKGROUND AND NOTATION

We consider functions of n d-valued inputs with one r-valued output. Let $R = \{0, 1, \ldots, r-1\}$ and $D = \{0, 1, \ldots, d-1\}$ be sets of r and d logic values respectively, where $r \geq 2$ and $d \geq 2$. Let $X = \{x_1, x_2, \cdots, x_n\}$ be a set of n variables, where x_i takes on values from D. A function $f(X)$ is a mapping $f : D^n \rightarrow R$. If X is a single variable x, $f(x)$ is represented as an d-tuple, $<f(0), f(1), \cdots, f(d-1)>$. For example, if $d = r = 4$, then $f(x) = <3, 2, 1, 0>$ represents a *complement* function in which 0 maps to 3, 1 to 2, 2 to 3, and 3 to 0. Let $U^r_{d,n}$ be the set of all r-valued functions of n d-valued inputs. Let $c(f)$, the *cost* of function f, be a mapping $c : U^r_{d,n} \rightarrow \boldsymbol{R}^{0+}$, where $\boldsymbol{R}^{0+}$ is the set of nonnegative real numbers. The cost function $c(f)$ introduced in [6] for the design of 4-valued CCD logic circuits correlates with the chip area occupied by the most compact implementation of f.

In the realization of a given function by costtable, costtable functions are combined using a connecting operation. The connecting operation $+$ between functions used in this paper is similar to ordinary addition with logic values viewed as integers. That is, if $f(X)$ is represented as the sum $f(X) = f_1(X) + f_2(X) + \cdots + f_m(X)$, then, for any assignment v of values to X, $f(v) = f_1(v) + f_2(v) + \cdots + f_m(v)$, where each $f_i(v)$ is taken as an integer and where + is ordinary addition, except when the sum exceeds $r - 1$, the highest output logic value, in which case $+$ is undefined. For example, if $f_1(x) = <0, 1, 2, 3>$ and $f_2(x) = <3, 2, 1, 0>$, then $f_1(x) + f_2(x) = <3, 3, 3, 3>$ and $f_1(x) + f_1(x)$ is undefined. The first example shows that the sum of the identity function $<0, 1, 2, 3>$ and the complement function $<3, 2, 1, 0>$ is the constant function $<3, 3, 3, 3>$.

Let s be the cost of realizing the sum of two functions. Thus, the cost of the realization $f = f_1 + f_2 + \cdots + f_m$ is $c(f_1) + c(f_2) + \cdots + c(f_m) + (m-1)s$, where the last term is the cost of $(m-1)$ two-input adders.

Function f is a *basis function* iff $f(X)$ is 1 for exactly one assignment of values to X and is 0 otherwise. Let BT be the set of all basis functions plus the function that is 0 for all assignments of values to the variables (e.g., <0,0,0,0>). BT is called the *basis costtable.* F is a *costtable* iff $BT \subseteq F \subseteq U^r_{d,n}$. The condition $BT \subseteq F$ guarantees that all functions can be realized as the sum $+$ of costtable functions. For example, in $U^4_{4,1}$ $BT = \{<0, 0, 0, 0>, <0, 0, 0, 1>, <0, 0, 1, 0>, <0, 1, 0, 0>, <1, 0, 0, 0>\}$. If $<0, 0, 0, 1>$ is missing, it is impossible to realize certain functions, including $<0, 0, 0, 1>$ itself. Conversely, any function $<a_0, a_1, a_2, a_3>$ can be realized as the sum of functions exclusively from BT, specifically a_0 functions of the form $<1, 0, 0, 0>$, a_1 $<0, 1, 0, 0>$, a_2 $<0, 0, 1, 0>$, and a_3 $<1, 0, 0, 0>$.

$c_F(f)$, the cost of realizing $f \in U^r_{d,n}$ with respect to costtable F is the minimal cost realization, specifically

$$c_F(f) = \min_{f_1, f_2, \cdots, f_m \in F} \{c(f_1) + c(f_2) + \cdots + c(f_m) + (m-1)s\},$$

where c is the the function. $f = f_1 + f_2 + \cdots + f_m$ is said to be a *minimal cost realization* of f. The *total cost*, $T(F)$, of costtable F is

$$T(F) = \sum_{f \in U^r_{d,n}} c_F(f).$$

In the context of a completely specified cost function, we consider the concept of a minimal costtable. F_t is a *minimal costtable* of size t if $|F_t| = t$ and $T(F_t) \le T(F)$, for all F, such that $|F| = t$. The term "minimal" describes the cost over *all* realizations of a costtable. That is, in a minimal costtable, certain minimal realizations may cost more than in some other costtable; however, the total cost of all realizations is no more.

Consider what happens when a function f is added to a costtable F. The resulting costtable will give realizations at least as good as the original costtable. If the added function f has a cost lower than the cost of any realization of f under F, then there is a nonzero reduction in the total cost. We define the *self-reduction* of f to be

$$\Gamma(f) \equiv \max\{c_{BT}(f) - c(f), 0\}.$$

That is, the self-reduction is the reduction in the cost of f obtained by the addition of f to the basis costtable, BT. If $c_{BT}(f) > c(f)$, there is a benefit to adding f to BT. If the cost function is such that $c_{BT}(f) \le c(f)$, then a minimal realization of f in the costtable $BT \cup \{f\}$ uses only basis functions, and there is no advantage to adding f to BT. Further, for any costtable F, $c_F(f) \le c_{BT}(f)$, since $BT \subseteq F$. If $c(f)$ is lower than the cost of *any* combination of other functions, then adding f to F reduces the cost of f, regardless of the composition of F.

III. COMPLEXITY OF THE COSTTABLE REALIZATION PROBLEM

The *Costtable Realization* (CR) problem is:

> Given a costtable F, a function f, and a cost function c, find a minimal cost realization $f = f_1 + f_2 + \cdots + f_m$, where $f_i \in F$.

The *Costtable Decision* (CD) problem is:

> Given a costtable F, a function f, a cost function c, and a target cost P, does there exist a realization $f = f_1 + f_2 + \cdots + f_m$ such that $c(f_1 + f_2 + \cdots + f_m) \le P$, where $f_i \in F$?

An instance of this problem is denoted as CD(F, f, c, P). CD(F, f, c, P) is said to be satisfied iff such a realization exists. We are interested in how the complexity of the Costtable Decision problem varies with $|F|$, the *size* of an instance of CD(F, f, c, P).

Consider the following solution to CD(F, f, c, P). Using i to index the functions in F, let k_i be the number of times the ith function in F is used in the realization of a given function f. Since the largest logic value in f is $r-1$, $k_i \le r-1$. To form a realization, one can scan F, nondeterministically choosing k_i, sum as many as $r-1$ copies of each costtable function for each possible d^n assignments of values to inputs taking no more than $\mathbf{O}((r-1)d^n \mid F \mid)$ time, check whether the chosen function is a realization of f taking $\mathbf{O}(d^n)$ time, and check whether the cost is less than or equal to P in $\mathbf{O}((r-1) \mid F \mid)$ time. The proves the following.

Lemma 1: There exists a non-deterministic algorithm that solves CD(F, f, c, P) in time polynomial in the size of F.

It is convenient to couch the Costtable Realization problem as a maximization problem. The following lemma shows how this is done.

Lemma 2: Given a function f and a costtable F, let $f_1 + f_2 + \cdots + f_m$ be a minimal cost realization of f with respect to F. Then, the cost of this realization, $c_F(f_1 + f_2 + \cdots + f_m)$, is $c_{BT}(f) - \sum_{i=1}^{m} \Gamma(f_i)$, where $c_{BT}(f)$ is the cost of f using the basis costtable and $\Gamma(f_i)$ is the self-reduction of f_i.

Proof: Since $f_1 + f_2 + \cdots + f_m$ is a minimal cost realization,

$$c_F(f_1 + f_2 + \cdots + f_m) = c(f_1) + c(f_2) + \cdots + c(f_m) + (m-1)s.$$

There is exactly one way to realize f using the basis costtable. These functions can be grouped to realize $f_1, f_2, \ldots$, and f_m, and we can write,

$$c_{BT}(f) = c_{BT}(f_1) + c_{BT}(f_2) + \cdots + c_{BT}(f_m) + (m-1)s.$$

Subtracting yields

$$c_F(f_1 + f_2 + \cdots + f_m) - c_{BT}(f) = \sum_{i=1}^{m} \Big[c(f_i) - c_{BT}(f_i) \Big].$$

Since $f_1 + f_2 + \cdots + f_m$ is a minimal cost realization of f, a combination of basis functions that realizes f_i has a cost no less than $c(f_i)$, and $c(f_i) \le c_{BT}(f_i)$. Therefore, $c_{BT}(f_i) - c(f_i) = \Gamma(f_i)$, and so,

$$c_F(f_1 + f_2 + \cdots + f_m) - c_{BT}(f) = -\sum_{i=1}^{m} \Gamma(f_i).$$

Adding $c_{BT}(f)$ to each side completes the proof.

Q.E.D.

Rather than minimize $c_F(f)$, we choose to maximize $\sum_{i=1}^{m} \Gamma(f_i)$, where $f = f_1 + f_2 + \cdots + f_m$ and $f_i \in F$. Viewed this way, the CD problem is similar to the Knapsack Decision problem, which is known to be NP-complete [3].

The *Knapsack Decision* problem can be stated as follows:

Given a set Q of objects, a size function $s: Q \to \mathbf{Z}^+$, a value function $v: Q \to \mathbf{Z}^+$, a size S, and a value V, is there a subset $Q' \subseteq Q$ such that $\sum_{u \in Q'} v(u) \ge V$ and

$\sum_{u \in Q'} s(u) \le S$, where Z^+ is the set of positive integers?

We denote an instance of the problem as KD(Q, s, v, S, V). KD(Q, s, v, S, V) is said to be satisfied iff such a subset Q' exists.

Definition: Let Φ be a transformation from any instance of the Knapsack Decision problem to an instance of the Costtable Decision problem

$$\Phi(\mathrm{KD}(Q, s, v, S, V)) = \mathrm{CD}(F, f, c, P),$$

with F, f, c, and P defined as follows:

1) F is a costtable over the universe of one variable d-valued input, r-valued output functions, $U^r_{d,1}$, where $r = S+1$ and $d = |Q|+1$. Let u_i be the ith element in an ordering of the elements of Q. Let F contain the d basis functions, the function mapping to all 0's ($<0, 0, \cdots, 0>$), and $d-1$ non-basis functions $f_1, f_2, \ldots, f_{d-1}$, where f_i corresponds to the ith element in Q. Let f_i be of the form $f_i(0) = s(u_i)$, $f_i(i) = 1$, and $f_i(j) = 0$ for $1 \le j \le d-1, j \ne i$. We have

$$\begin{aligned} f_1 &= <s(u_1), 1, 0, 0, \cdots, 0> \\ f_2 &= <s(u_2), 0, 1, 0, \cdots, 0> \\ &\ \vdots \qquad\qquad \vdots \\ f_{d-1} &= <s(u_{d-1}), 0, 0, 0, \cdots, 1>. \end{aligned}$$

Thus, $F = BT \cup \{f_i \mid 1 \le i \le d-1\}$.

2) Function f has the form $f = <S, 1, 1, 1, \cdots, 1>$. Note that each f_i can be used at most one time in the realization of f, since $f(i) = 1$ for $1 \le i \le d-1$. This corresponds to the limitation of using each element $u \in S$ at most once in the Knapsack Decision problem. Also, since $f(0) = S$, the sum $\sum f_i(0)$ over the f_i's used in a realization of f (i.e. $s(u_i)$) must be less than or equal to S.

3) Let $c(f_i) = s(u_i)$ for $1 \le i \le d-1$. Let the cost of combining two functions be 1. The cost of basis function b_j, is defined as follows.

$$c(b_j) = \begin{cases} 0 & \text{if } j = 0 \\ v(u_j) & \text{otherwise} \end{cases},$$

where $b_j(j) = 1$ and $b_j(i) = 0$ for $i \ne j$. That is, the cost of $<1, 0, \cdots, 0>$ is 0, while the cost of all other basis functions is the value of some object in Q. The cost of the constant function $<0, 0, \cdots, 0>$ is 0.

4) P is defined by

$$P = \sum_{u_i \in Q} v(u_i) - V + (S + d - 2). \tag{1}$$

Example: Consider a knapsack defined as follows. Let $Q = \{u_1, u_2, u_3, u_4\}$, and let s and v have the values

	$s(u_i)$	$v(u_i)$
u_1	3	4
u_2	3	4
u_3	2	3
u_4	2	2

Let $S = 6$ and $V = 7$.

Of the 16 ways to choose subsets of Q, there are three that satisfy $KD(Q, s, v, S, V)$, as shown below.

$Q_1 = \{u_1, u_2\}$	$\sum_{u \in Q_1} v(u) = 8 \geq V = 7$ $\sum_{u \in Q_1} s(u) = 6 \leq S = 6$
$Q_2 = \{u_1, u_3\}$	$\sum_{u \in Q_2} v(u) = 7 \geq V = 7$ $\sum_{u \in Q_2} s(u) = 5 \leq S = 6$
$Q_3 = \{u_2, u_3\}$	$\sum_{u \in Q_3} v(u) = 7 \geq V = 7$ $\sum_{u \in Q_3} s(u) = 5 \leq S = 6$

Applying the transformation yields $r = 7$ and $d = 5$. That is, the costtable consists of 7-valued functions on a 5-valued variable. The costtable functions are

Function	Cost	
<0,0,0,0,0>	0	0
<1,0,0,0,0>	0	0
<0,1,0,0,0>	4	$v(u_1)$
<0,0,1,0,0>	4	$v(u_2)$
<0,0,0,1,0>	3	$v(u_3)$
<0,0,0,0,1>	2	$v(u_4)$
<3,1,0,0,0>	3	$s(u_1)$
<3,0,1,0,0>	3	$s(u_2)$
<2,0,0,1,0>	2	$s(u_3)$
<2,0,0,0,1>	2	$s(u_4)$

The function to be synthesized is $f = $ <6,1,1,1,1>, and $P = 15$. The instance of the costtable decision problem, $CD(F, f, c, P)$ so formed is satisfied by exactly three realizations of f as follows.

Function	Cost	Function	Cost	Function	Cost
<3,1,0,0,0>	3	<3,1,0,0,0>	3	<3,0,1,0,0>	3
<3,0,1,0,0>	3	<2,0,0,1,0>	2	<2,0,0,1,0>	2
<0,0,0,1,0>	3	<0,0,1,0,0>	4	<0,1,0,0,0>	4
<0,0,0,0,1>	2	<0,0,0,0,1>	2	<0,0,0,0,1>	2
		<1,0,0,0,0>	0	<1,0,0,0,0>	0
Adders	3	Adders	4	Adders	4
Total	14	Total	15	Total	15

These three realizations match left to right with $\{u_1, u_2\}$, $\{u_1, u_3\}$, and $\{u_2, u_3\}$, the subsets satisfying KD(Q, s, v, S, V). Note that, of the three realizations of <6,1,1,1,1,1>, one is uniquely minimal, that given in the left hand column above.

We can make the following general statement.

Lemma 3: Φ is a polynomial time reduction of the Knapsack Decision problem to the Costtable Decision problem such that KD(Q, s, v, S, V) is satisfied iff CD(F, f, c, P) = Φ(KD(Q, s, v, S, V)) is satisfied.

Proof: The proof is divided into three parts. First, it is shown that Φ takes polynomial time. Then it is shown that if KD(Q, s, v, S, V) is satisfied then Φ(KD(Q, s, v, S, V)) is satisfied (only if). Finally, it is shown that if Φ(KD(Q, s, v, S, V)) is satisfied then KD(Q, s, v, S, V) is satisfied (if).

To form the costtable $F \subseteq U_{d,1}^r$, Φ generates $d-1 = |Q|$ non-basis functions, d basis functions, and the constant function $<0,0, \cdots ,0>$. Each function can be described by a truth table with $d = |Q| + 1$ entries. An entry in the truth table can be made in constant time. Thus, the total time to generate F is $\mathbf{O}(|Q|^2)$. A cost is then assigned to each of these functions requiring constant time per function. Since $s(u_i)$ can be computed in constant time, the target function f can be formed in $\mathbf{O}(|Q|)$ time. Finally, P requires the summation of all $v(u_i)$, which also takes $\mathbf{O}(|Q|)$ time. Since each step takes at most polynomial time, the entire transformation takes polynomial time.

In preparation for the second two parts, let

$$g_i = \begin{cases} f_i & \text{if } u_i \in Q' \text{ and } 1 \le i \le d-1 \\ b_i & \text{if } u_i \notin Q' \text{ and } 1 \le i \le d-1 \\ b_0 & \text{if } d \le i \le m \end{cases}$$

where $m = S - S' + |Q|$, $S = \sum_{u_i \in Q} s(u_i)$, and $S' = \sum_{u_i \in Q'} s(u_i)$. We now show that $g_1 + g_2 + \cdots + g_m = f$. Consider $g_1 + g_2 + \cdots + g_m$ when the input value is 0.

$$\sum_{i=1}^{m} g_i(0) = \sum_{u_i \in Q'} f_i(0) + \sum_{u_i \notin Q'} b_i(0) + \sum_{i=d}^{m} b_0(0)$$

$$= \sum_{u_i \in Q'} s(u_i) + 0 + (m-d+1) = S' + 0 + (S-S') = f(0).$$

When the input value is not 0, $g_1 + g_2 + \cdots + g_m$ is evaluated as follows. By the definition of f_i and b_i, $g_i(j) = 0$ if $i \neq j$ and $1 \leq j$. Therefore, $\sum_{i=1}^{m} g_i(j) = 1 = f(j)$, for $1 \leq j \leq d-1$. This proves that $g_1 + g_2 + \cdots + g_m = f$.

The cost of realization $f = g_1 + g_2 + \cdots + g_m$ is

$$\sum_{u_i \in Q'} s(u_i) + \sum_{u_i \notin Q'} v(u_i) + 0 + (m-1)$$

or

$$S' + \sum_{u_i \in Q} v(u_i) - V' + (S - S' + |Q| - 1)$$

From (1), the cost of this realization is $P - V' + V$.

(only if) Assume KD(Q, s, v, S, V) is satisfied by Q'. The *size* of this collection is $S' = \sum_{u_i \in Q'} s(u_i)$, and the *value* is $V' = \sum_{u_i \in Q'} v(u_i)$. Since Q' satisfies KD(Q, s, v, S, V), $S' \leq S$ and $V' \geq V$. Now consider $c_F(f)$, the minimal cost realization of f in costtable F. Because the cost of the realization $g_1 + g_2 + \cdots + g_m$ is an upper bound on the minimal cost realization, $c_F(f) \leq P - V' + V$. Since $V' \geq V$, $c_F(f) \leq P$. Thus, the Costtable Decision problem is satisfied.

(if) Assume $\Phi(\text{KD}(Q, s, v, S, V)) = \text{CD}(F, f, c, P)$ is satisfied by the realization $f = h_1 + h_2 + \cdots + h_l$, where $h_i \in F$. Then, $\sum_{i=1}^{l} c(h_i) + (l-1) \leq P$. We show that the Knapsack Decision problem is satisfied for

$$Q' = \{u_i \mid h_i \notin BT\}.$$

To calculate the "size" of the solution, consider the function evaluated at 0.

$$\sum_{i=1}^{m} h_i(0) = f(0)$$

$$\sum_{u_i \in Q'} h_i(0) + \sum_{u_i \notin Q'} h_i(0) = S$$

Since $h_i(0) \geq 0$,

$$\sum_{u_i \in Q'} h_i(0) \leq S$$

and

$$\sum_{u_i \in Q'} s(u_i) \leq S.$$

To calculate the "value" of the solution, consider the cost of the realization $f = h_1 + h_2 + \cdots + h_l$. Because this is a solution to CD(F, f, c, P),

$$\sum_{i=1}^{l} c(h_i) + (l-1) \leq P.$$

Inserting the definitions of P and $c(h_i)$ into this equation yields,

$$\sum_{u_i \in Q'} s(u_i) + \sum_{u_i \notin Q'} v(u_i) + l - 1 \leq \sum_{u_i \in Q} v(u_i) - V + (S + d - 2).$$

Rearranging, yields

$$V + \left[l - [(d-1) + S - \sum_{u_i \in Q'} s(u_i)] \right] \leq \sum_{u_i \in Q'} v(u_i).$$

We show that the term in large brackets is 0. Thus, $V \leq \sum_{u_i \in Q} v(u_i)$, and so the Knapsack decision problem has a solution. Each of the $d - 1$ 1 terms in $f = <S, 1, 1, \cdots, 1>$ is realized by either a b_i or an f_i, for $1 \leq i \leq d-1$. The f_i terms contribute $\sum_{u_i \in Q'} s(u_i)$ to $f(0)$. Thus, $S - \sum_{u_i \in Q'} s(u_i)$ copies of b_0 are needed. It follows that $l = (d-1) + S - \sum_{u_i \in Q'} s(u_i)$. Thus, a solution to $\mathrm{KD}(Q, s, v, S, V)$ exists such that $\sum_{u_i \in Q'} s(u_i) \leq S$ and $\sum_{u_i \in Q'} v(u_i) \geq V$.

Q.E.D.

This allows us to state the main result.

Theorem 1: The Costtable Decision problem is NP-complete.

IV. CONCLUDING REMARKS

The fundamental nature of the costtable technique has inspired this examination of its computational complexity. For example, the design of an instruction set for a processor is analogous to the costtable design technique. The question of whether to include simple or complex instructions, as in the question of a RISC versus a CISC system, is similar to deciding whether to include simple, low-cost functions or complex high-cost functions is a costtable for logic design. We show that the latter problem is NP-complete. The proof rests on showing a transformation from the Knapsack Decision Problem. This confirms our observation that the design of a costtable is complex.

REFERENCES

[1] M. H. Abd-El-Barr, T. D. Hoang, and Z. G. Vranesic, "The incremental-cost approach for synthesis of CCD 4-valued unary functions," *Proc. of the 18th Inter. Symp. on Multiple-Valued Logic*, May 1988.

[2] J. T. Butler and K. A. Schueller, "On the equivalence of cost functions in the design of circuits by costtables", *IEEE Transactions on Computers*, C-39, June 1990, pp. 842-844.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness,* W. H. Freeman and Company, 1979.

[4] J.F. Gimpel, "A method of producing a boolean function having an arbitrarily prescribed prime implicant table," *IEEE Trans. on Electron. Comput.*, June 1965, pp. 485-488.

[5] R.M. Karp, "Reducibility among combinatorial problems," in R. E. Miller and J. W. Thatcher, *Complexity of Computer Computations*, Plenum Press, 1972, pp. 85-103.

[6] H. G. Kerkhoff and H. A. J. Robroek, "The logic design of multiple-valued logic functions using charge-coupled devices," *Proc. of the 12th Inter. Symp. on Multiple-Valued Logic*, May 1982, pp. 35-44.

[7] S. Onneweer and H. J. Kerkhoff, "High-radix current-mode CMOS circuits based on the truncated difference operator," *Proc. of the 17th Inter. Symp. on Multiple-Valued Logic*, May 1987, pp. 188-195.

[8] K. A. Schueller, P. P. Tirumalai, and J. T. Butler, "An analysis of the costtable approach to the design of multiple-valued circuits," *Proc. of the 16th Inter. Symp. on Multiple-Valued Logic*, May 1986, pp. 42 - 50.

On Finding Obstructions for Layout Permutation Problems*

Nancy G. Kinnersley
Department of Computer Science
University of Kansas
Lawrence, KS 66045-2192

Abstract

Powerful nonconstructive graph theory results have been used to prove that the fixed-parameter variants of several $\mathcal{NP}$-complete layout permutation problems have polynomial-time decision algorithms. Each such algorithm depends on the existence of a finite but unknown obstruction set. In this paper we show how problem-specific methods can be used to identify some obstructions and how constructive methods used to identify minor-minimal obstructions for the VERTEX SEPARATION problem can also be used to find immersion-minimal obstructions for the MIN CUT LINEAR ARRANGEMENT problem.

1. Introduction

GATE MATRIX LAYOUT [3], VERTEX SEPARATION [14], MIN CUT LINEAR ARRANGEMENT [10], and NODE SEARCH NUMBER [13] are $\mathcal{NP}$-complete combinatorial problems with important applications to VLSI design. Recently, Robertson-Seymour poset techniques have been used to *nonconstructively* establish the existence of $O(n^2)$ decision algorithms for the fixed parameter variants of these problems. (See, for example, [6–8] for applications of tools from [17–20].) An instance of a problem is modeled as a graph, and the decision algorithm determines whether a given encoding is a "yes" or a "no" instance by determining if it contains an element of a finite basis of forbidden graphs (the obstruction set). The underlying theory tells us only that the set is finite. It does not tell how to identify all members of such a set, the cardinality of the set, or even the order of the largest graph in the set.

In this paper we examine ways of isolating elements of obstruction sets for the problems above. We describe characteristics of both obstructions and layouts, and show how a number of obstructions can be constructively obtained. In the next section, we discuss relevant background information. In Section 3, we examine ways of obtaining elements of obstruction sets based on both the minor and immersion orders by comparing the VERTEX SEPARATION and MIN CUT LINEAR ARRANGEMENT problems. The final section summarizes our results.

2. Background

A graph H is less than or equal to a graph G in the *minor* order, written $H \leq_m G$, if and only if a graph isomorphic to H can be obtained from G by a series of these two operations: taking a subgraph and contracting an arbitrary edge. Figure 1 illustrates this operation.

---

* This research has been supported in part by University of Kansas General Research allocation #3800-20-0038 and by the National Science Foundation under grant CCR-9008725.

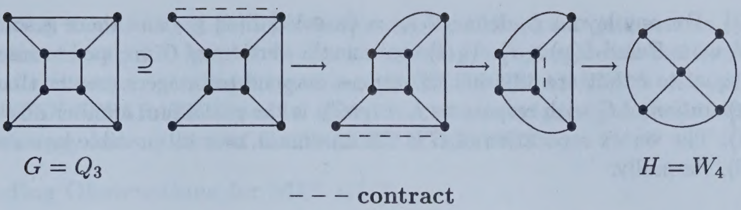


Figure 1: Illustration showing W_4 is a minor of Q_3

A family F of graphs is said to be *closed* under the minor order if the facts that G is in F and that $H \leq_m G$ together imply that H must be in F. The *obstruction set* for a family F of graphs is the set of graphs in the complement of F that are minimal in the minor order. Therefore, if F is closed under the minor order, it has the following characterization: G is in F if and only if $H \not\leq_m G$ for every H in the obstruction set for F.

Theorem 2.1 [20] Any set of finite graphs contains only a finite number of minor-minimal elements.

Theorem 2.2 [19] For every fixed graph H, the problem that takes as input a graph G and determines whether $H \leq_m G$ is solvable in polynomial time.

A graph H is less than or equal to a graph G in the *immersion* order, written $H \leq_i G$, if and only if a graph isomorphic to H can be obtained from G by a series of these two operations: taking a subgraph and *lifting* pairs of adjacent edges. Adjacent edges uv and vw are *lifted* by replacing them with the edge uw. Figure 2 illustrates this operation. The relation $\leq_i$, like $\leq_m$, defines a partial ordering on graphs with the associated notions of closure and obstruction sets. In order to keep our terminology analogous to that for the minor order, we say that H is an *immersion* of G if $H \leq_i G$. The notation $H <_i G$ means that H is a proper immersion of G.

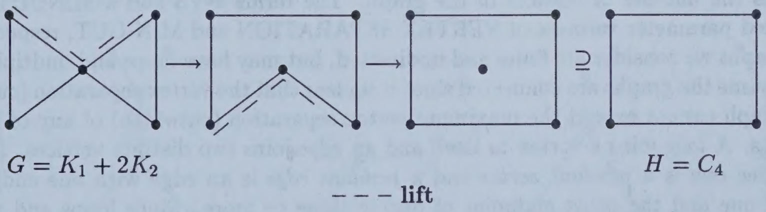


Figure 2: Illustration showing C_4 is an immersion of $K_1 + 2K_2$

The following analogs of Theorems 2.1 and 2.2 guarantee the existence of polynomial-time decision algorithms for any immersion closed family of graphs.

Theorem 2.3 [18] Any set of finite graphs contains only a finite number of immersion-minimal elements.

Theorem 2.4 [8] For every fixed graph H, the problem that takes as input a graph G and determines whether $H \leq_i G$ is solvable in polynomial time.

A *(linear) layout* of a graph $G = \langle V, E\rangle$ is a one to one mapping $L : V \to \{1,$

2,…,$|V|$}. For any layout L, define $V_L(i) = \{u \in V \mid L(u) \leq i$ and there is some $v \in V$ such that $uv \in E$ and $L(v) > i\}$. $V_L(i)$ contains the vertices of G mapped to integers less than or equal to i that are adjacent to vertices mapped to integers greater than i. The *vertex separation* of G with respect to L, $vs_L(G)$, is the maximum number of vertices in any $V_L(i)$. The vertex separation of G is the minimum, over all possible layouts L of G, of $vs_L(G)$. Formally,

$$vs_L(G) = \max_{1 \leq i < |V|} \{|V_L(i)|\}$$

and

$$vs(G) = \min\{vs_L(G) \mid L \text{ is a linear layout of } G\}$$

The VERTEX SEPARATION problem [14] asks, for input graph G and positive integer k, whether $vs(G) \leq k$.

The *cutwidth* of a graph G under layout L, $cw_L(G)$, is the maximum number of edges of G cut by a vertical line drawn between consecutive vertices. The cutwidth of G, $cw(G)$, is the minimum over all possible layouts L of $cw_L(G)$. Formally,

$$cw_L(G) = \max_{1 \leq i < |V|} |\{uv \in E \mid L(u) \leq i < L(v)\}|$$

and

$$cw(G) = \min\{cw_L(G) \mid L \text{ is a linear layout of } G\}$$

The MIN CUT LINEAR ARRANGEMENT problem [10] (hereafter called MIN CUT) asks, for input graph G and positive integer k, whether $cw(G) \leq k$.

Both VERTEX SEPARATION and MIN CUT are $\mathcal{NP}$-complete even when restricted to planar graphs with maximum vertex degree three [16], but can be solved in polynomial time for trees [4, 21]. For fixed k, the set of graphs with vertex separation at most k is closed under the minor order, and the family of graphs with cutwidth less than or equal to k is closed under the immersion order [8]. Therefore, Theorems 2.1–2.4 guarantee that the fixed parameter variants of these problems have polynomial-time decision algorithms based on obstruction sets. The time bound for each such algorithm is $O(n^2)$ where n denotes the number of vertices in the graph. The terms k-VS and k-MINCUT denote the fixed parameter variants of VERTEX SEPARATION and MIN CUT, respectively.

Graphs we consider are finite and undirected, but may have loops and multiple edges. We assume the graphs are connected since it is clear that the vertex separation (cutwidth) of a graph cannot exceed the maximum vertex separation (cutwidth) of any of its components. A *loop* joins a vertex to itself and an *edge* joins two distinct vertices. A vertex of degree one is a *pendant vertex* and a *pendant edge* is an edge with one endpoint of degree one and the other endpoint of degree three or more. Since loops and multiple edges do not increase a graph's vertex separation, we consider only simple graphs for that problem.

The combinatorial problem GATE MATRIX LAYOUT [3] is the archetypical example used by Fellows and Langston to illustrate the application of Robertson and Seymour poset techniques [6–8]. It was shown in [11] that the vertex separation of a graph equals its path-width (see [17] for a definition); [13] proved that the node search number of a graph is one more than its vertex separation; and [9] showed that the gate matrix layout cost of a graph is one more than its path-width. Thus, the VERTEX SEPARATION, GATE MATRIX LAYOUT, and NODE SEARCH NUMBER problems are the same. Kinnersley and Langston [12] used a combination of mathematical proof and exhaustive case-checking to prove that the obstruction set for three-track gate matrix layout (2-VS) contains exactly 110 graphs. In the next section, we compare obstruction isolation methods for the two linear layout problems VERTEX SEPARATION and MIN CUT. The VERTEX SEPARATION results stated below were originally derived in [12] using the terminology and notation of GATE MATRIX LAYOUT.

3. Finding Obstructions for MIN CUT

While the size of obstruction sets for the VERTEX SEPARATION problem grows extremely rapidly as a function of the parameter k, empirical evidence suggests that only a relatively small number of obstructions are needed in most applications [12]. Thus, herein we do not attempt to identify all elements of obstruction sets, but focus instead on how several graphs in each of the k-MINCUT obstruction sets can be readily identified. In addition to problem specific methods for MIN CUT, we show how constructive methods used to identify minor-minimal obstructions for k-VS can also be used to find immersion-minimal obstructions for k-MINCUT.

An obstruction for VERTEX SEPARATION or MIN CUT is a graph that represents a "no" instance for parameter k (it has no cost k layout) and that is minimal for parameter k (each of its proper minors (immersions) does have a cost k layout). For 1-VS, the only obstructions are K_3 and $S(K_{1,3})$[1] [1, 6]. For 1-MINCUT the only obstructions are $K_{1,3}$ and the one vertex graph with a single loop.

In [15] it was shown that reducing a graph by lifting the edges incident on a vertex of degree two does not change its cutwidth. Thus, no MIN CUT obstruction contains a vertex of degree two, and we hereafter assume all graphs have been reduced in this way. The cutwidth of a graph with one or more loops is the same as that of the graph in which a single degree two vertex has been inserted into each loop. Figure 3 depicts the five forbidden components for graphs with cutwidth two identified in [15]. The order relation used there is the *weak immersion order* [5], so while the graphs in Figure 3 are immersion minimal obstructions for 2-MINCUT, there are additional graphs in that obstruction set. As we show below, each of the graphs in Figure 3 can be generalized to produce one or more obstructions for any value of k.

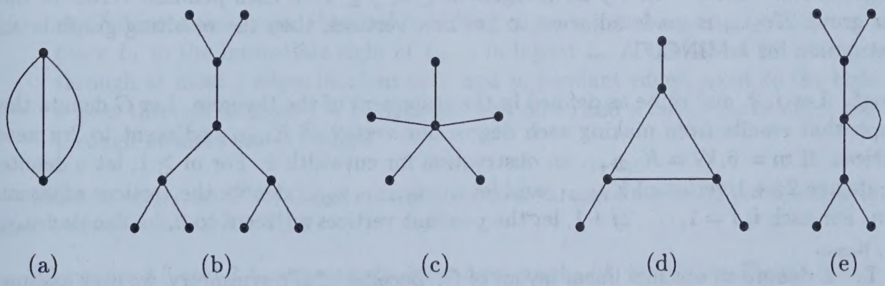


Figure 3: Forbidden Components for Graphs with Cutwidth 2 [15]

It is easy to see that, for either problem, inserting a new edge in a graph can increase the cost of a layout, and hence the cost of the graph by at most one. k-MINCUT differs from k-VS in that any graph with a vertex of degree greater than $2k$ represents a "no" instance. For $k \geq 1$, the star graph $K_{1,2k+1}$ is an obstruction for k-MINCUT, but has vertex separation 1. Since an obstruction for k-VS has vertex separation $k+1$, the obstruction sets for different values of the parameter are disjoint. However, for even k, the obstruction sets for k- and $(k+1)$-MINCUT have at least one graph in common, namely, the graph with a single vertex and $k+1$ loops.

---

[1] $S(K_{1,3})$ is the graph obtained by subdividing each edge of $K_{1,3}$.

Let L denote any layout of K_n, the complete graph on n vertices. Clearly, K_n has vertex separation $n-1$ and is an obstruction for $(n-2)$-VS. The number of edges cut by a line drawn between consecutive vertices in L is the product of the number of vertices to the left of the line and the number to its right. Thus, the cut of maximum size occurs in the "middle" of a layout and has cost $\lceil \frac{n}{2} \rceil \cdot \lfloor \frac{n}{2} \rfloor$. We now show that K_n is an immersion-minimal obstruction. For ease of discussion, we hereafter call a vertical line drawn between consecutive vertices in a layout a *cut*. A layout will be given as either a function or as a sequence of vertices.

Theorem 3.1. For $n > 3$, K_n is an obstruction for cutwidth $\lceil \frac{n}{2} \rceil \cdot \lfloor \frac{n}{2} \rfloor - 1$.

Proof. From the discussion above, it is clear that K_n has cutwidth $\lceil \frac{n}{2} \rceil \cdot \lfloor \frac{n}{2} \rfloor$. The graph with a single loop, an obstruction for 1-MINCUT, is a proper immersion of K_3. For $n > 3$, let $v_1, v_2, \ldots, v_n$ denote the vertices of K_n. Since all layouts of K_n are equivalent, let $L(v_i) = i$, for $1 \leq i \leq n$. Due to the symmetry of K_n, to prove minimality we need only show that removing edge v_1v_n, or lifting adjacent edges v_1v_{n-1} and v_1v_n leaves a graph with cutwidth less than that of K_n. If $G' = K_n \setminus \{v_1v_n\}$, then the cost of every cut in L decreases by one and $cw(G') < cw(K_n)$. If edges v_1v_{n-1} and v_1v_n are lifted, the cost of every cut in L, except the one between v_{n-1} and v_n, decreases by two, so the resulting graph has cutwidth less than that of K_n. Therefore, if $n > 3$, K_n is an immersion-minimal obstruction for cutwidth $\lceil \frac{n}{2} \rceil \cdot \lfloor \frac{n}{2} \rfloor - 1$. □

For $k \geq 2$, the graph on two vertices with $k+1$ edges is an obstruction for k-MINCUT. (If $k = 1$, the graph reduces to a loop.) This is the generalization of graph (a) in Figure 3. Theorem 3.2 shows how some tree obstructions for k-MINCUT, including graphs (b) and (c) in Figure 3, can be found.

Theorem 3.2. Let $k = m + j$ for integers $m \geq 0$, $j \geq 1$. If each pendant vertex of the star graph $K_{1,2j+1}$ is made adjacent to $2m$ new vertices, then the resulting graph is an obstruction for k-MINCUT.

Proof. Let j, k, and m be as defined in the statement of the theorem. Let G denote the graph that results from making each degree one vertex of $K_{1,2j+1}$ adjacent to $2m$ new vertices. If $m = 0$, $G = K_{1,2k+1}$, an obstruction for cutwidth k. For $m \geq 1$, let v denote the degree $2j+1$ vertex of $\mathrm{K}_{1,2j+1}$, and let $u_1, u_2, \ldots, u_{2j+1}$ denote the vertices adjacent to v. For each i, $i = 1, \ldots, 2j+1$, let the pendant vertices adjacent to u_i be denoted $u_{i,1}$, $\ldots$, $u_{i,2m}$.

Let L denote an optimal linear layout of G. Because of G's symmetry, we may assume that $L(u_i) < L(u_{i+1})$ for each i, $1 \leq i \leq 2j$. Let c denote the cut immediately to the right of u_{j+1}. Since u_{j+1} has degree $2m+1$, we assume, without loss of generality, that c passes through $m+1$ edges incident on u_{j+1}. If $L(v) < L(u_{j+1})$, then c also passes through the j edges $u_{j+2}v, \ldots, u_{2j+1}v$ making the total number of edges cut by c at least $j + m + 1 = k + 1$. If $L(v) > L(u_{j+1})$, then c passes through edges $u_1v, \ldots, u_jv$, again bringing the total number of edges cut to at least $k+1$. (Note that edge $u_{j+1}v$ is one of the $m+1$ edges incident on u_{j+1}.) Thus, in either case, $cw(G) \geq k+1$. For each i,

$i = 1, \ldots, 2j+1$, let G_i denote the component of $G \setminus \{v\}$ that contains vertex u_i and let L_i denote the following layout of G_i: $L_i = u_{i,1}, u_{i,2}, \ldots, u_{i,m}, u_i, u_{i,m+1}, \ldots, u_{i,2m}$. It is easy to verify that $L = L_1, L_2, \ldots, L_j, v, L_{j+1}, \ldots, L_{2j+1}$ is a layout of G with cutwidth $k+1$. Therefore, $cw(G) = k+1$.

To prove minimality, we show that an immersion of G resulting from the removal of an edge or lifting of a pair of edges has cutwidth $\leq k$. Because of G's symmetry, we need only consider the effects of removing an edge incident on u_1, and lifting a pair of edges one of which has u_1 as an endpoint. Layouts for these immersions of G will be modifications of the following linear layout: $L = L_2, L_3, \ldots, L_{j+1}, L', L_{j+2}, \ldots, L_{2j+1}$. The L_i, $1 \leq i \leq 2_{j+1}$ are as defined above, and layout L' will be given for each case. It is easy to check that a cut in any L_i passes through at most m pendant edges plus at most j edges incident on v. A cut between L_i's will pass through only edges incident on v. There are five immersions of G to consider:

(1) If $G' = G \setminus \{u_1u_{1,2m}\}$, let $L' = u_{1,1}, \ldots, u_{1,m}, u_1, v, u_{1,m+1}, \ldots, u_{1,2m-1}$. The cut to the immediate left of u_1 passes through k edges: j edges incident on v and m edges incident on u_1. The cut to the immediate right of u_1 passes through $j+1$ edges incident on v (including edge u_1v) and $m-1$ pendant edges incident on u_1.

(2) If $G' = G \setminus \{u_1v\}$, let $L' = v$ and place L_1 to the right of L_{2j+1} in layout L. Since v is not adjacent to u_1, no cut will pass through more than j edges incident on v and m pendant edges incident on some u_i.

(3) Let G' denote the graph obtained by lifting edges $u_1u_{1,1}$ and $u_1u_{1,2}$, and let $L' = u_{1,3}, \ldots, u_{m+1}, u_1, v, u_{1,m+2}, \ldots, u_{1,2m}, u_{1,1}, u_{1,2}$. Since u_1 is no longer adjacent to either $u_{1,1}$ or $u_{1,2}$, a cut in L' to the immediate left or right of u_1 passes through $m-1$ pendant edges incident on u_1, and at most $j+1$ edges incident on v.

(4) Let G' denote the graph obtained by lifting edges u_1v and $u_1u_{1,2m}$. Let $L' = v$, u_{2m}, and place vertices $u_{1,1}, \ldots, u_{1,m}\ u_1, u_{1,m+1}, \ldots, u_{1,2m-1}$, in that order, to the immediate right of L_{2j+1}. The cut to the immediate right of v has cost $j+1$, and all other cuts pass through at most m pendant edges and j edges incident on v.

(5) Let G' denote the graph obtained by lifting edges u_1v and $u_{2j+1}v$. Let $L' = v$, and place L_1 to the immediate right of L_{2j+1} in layout L. A cut to the left of v passes through at most j edges incident on v and m pendant edges, a cut to the right of v goes through at most $j-1$ edges incident on v, and a cut in L_1 or L_{2j+1} passes through at most $2m+1$ edges.

Since in each case $G' <_i G$ has cutwidth at most k, and since $cw(G) = k+1$, G is an obstruction for k-MINCUT. □

Theorems 3.3 and 3.4, respectively, show how graphs (d) and (e) in Figure 3 may be generalized to obtain obstructions for k-MINCUT.

Theorem 3.3. For $k \geq 2$, if each vertex of a triangle is made adjacent to $2k-3$ new vertices, then the resulting graph is an obstruction for k-MINCUT.

Proof Sketch. Let u, v, and w denote the vertices of a triangle, and let G denote the graph that results from making each of the vertices u, v, and w adjacent to $2k-3$ new (pendant) vertices. Let $u_1, u_2, \ldots, u_{2k-3}$ denote the pendant vertices adjacent to u, and label those adjacent to v and w analogously. Let L denote an optimal linear layout of G, and without loss of generality, assume $L(u) < L(v) < L(w)$. Since v has degree $2k-1$, some cut passes through $k+1$ edges: uw and k edges incident on v. Thus, $cw(G) \geq k+1$.

Let $L_u = u_1, \ldots, u_{k-1}, u, u_k, \ldots, u_{2k-3}$, $L_v = v_1, \ldots, v_{k-1}, v, v_k, \ldots, v_{2k-3}$, and $L_w = w_1, \ldots, w_{k-1}, w, w_k, \ldots, w_{2k-3}$. Since no cut in L_u, L_v, or L_w passes through more than $k-1$ edges, L_u, L_v, L_w is a linear layout of G with cost $k+1$, and $cw(G) = k+1$.

To see that G is minimal, we need to consider five ways of finding a proper immersion of G: (1) removal of a pendant edge, (2) removal of an edge of the triangle, (3) lifting two pendant edges (when $k > 2$), (4) lifting a pendant edge and an edge of the triangle, and (5) lifting two edges of the triangle. It is easy to verify that each of the immersions G' and layouts L' below has cutwidth at most k. For (1), let $G' = G \setminus \{vv_1\}$, let L'_v denote L_v with the first term removed, and let $L' = L_u, L'_v, L_w$. For (2), let $G' = G \setminus \{uw\}$, and let $L' = L_u, L_v, L_w$. For (3), let G' denote the graph obtained by lifting edges vv_1 and vv_2, and let $L' = L_u, L_v, L_w$. For (4), let G' denote the graph obtained by lifting edges uv and vv_1, and let $L' = L_u, L_v, L_w$. Finally, for (5), let G' denote the graph obtained by lifting edges uv and vw, and let $L' = L_u, w_1, \ldots, w_{k-2}, w, w_{k-1}, \ldots, w_{2k-3}, L_v$. □

Theorem 3.4. Let $k \geq 3$, and let $G_1 = G_2 = K_{1,2k-2}$ with v_1 and v_2 the degree $2k-2$ vertices of G_1 and G_2, respectively. Let G_3 denote the graph with vertex set $\{v, w\}$ and k edges between v and w. Then, $G = G_1 \cup G_2 \cup G_3 \cup \{vv_1, vv_2\}$ is an obstruction for k-MINCUT.

Proof Sketch. Let G_i, $1 \leq i \leq 3$, v_1, v_2, v, w and G be as defined in the statement of the theorem, and let L denote an optimal layout of G. If $L(v_1) < L(v) < L(v_2)$, then some cut passes through $k+1$ edges: the k edges between v and w and either vv_1 or vv_2. Therefore, without loss of generality, assume $L(v_1) < L(v_2) < L(v)$. If L maps $k-1$ of the pendant vertices adjacent to v_2 to integers greater than $L(v_2)$, then the cut to the immediate right of v_2 passes through $k+1$ edges: vv_1, vv_2, and $k-1$ pendant edges. The only other possibility is that the cut to the immediate left of v_2 passes through vv_1 and at least k pendant edges incident on v_2. Therefore, $cw(G) \geq k+1$. Any layout L' such that $L'(v_1) = k$, $L'(v_2) = 3k-1$, $L'(v) = 4k-1$ and $L'(w) = 4k$ that maps the pendant vertices adjacent to v_1 to integers less than $2k$ and maps the pendant vertices adjacent to v_2 to integers between $2k$ and $4k-2$, inclusive, is a cutwidth $k+1$ layout of G.

We omit the details of the proof that G is minimal. It is straightforward to construct cutwidth k layouts of the seven immersions that need to be considered: (1) removing one edge vw, (2) removing edge vv_1, (3) removing one pendant edge, (4) lifting two edges vw, (5) lifting vv_1 and vv_2, (6) lifting vv_1 and a pendant edge incident on v_1, and (7) lifting two pendant edges. □

Theorem 3.5. If G is an obstruction for k-MINCUT that contains two pendant edges incident on the same vertex, then the graph that results from replacing those edges by a loop has cost no more than G.

Proof Sketch. Let G denote an obstruction for k-MINCUT that contains pendant edges uv and vw. Let $G' = G \setminus \{uv\}$. Since G is an obstruction for k-MINCUT, $cw(G) \geq k+1$. Since G is minimal, $cw(G') \leq k$, and replacing edge vw by a loop on v brings the cost of the resulting graph to at most $k+1$. □

The replacement operation described in Theorem 3.5 can be used to obtain additional elements of the k-MINCUT obstruction sets. While we have not shown that this replacement guarantees the minimality of the resulting graph, it is easy to verify that the graphs obtained in this way from those in Figure 3 are immersion-minimal. Replacing pairs of pendant edges in the graphs in Figure 3 yields seven additional 2-MINCUT obstructions: three from (b), two from (c), and two from (e). Moreover, the graphs obtained by replacing one, two or all three of the pendant edges in graph (d) by loops, and the graph obtained by replacing the five pendant edges in (c) by a total of three loops also are 2-MINCUT obstructions. Thus the obstruction set for 2-MINCUT has at least 16 elements.

Theorems 3.6, 3.7, and 3.8 show three ways in which graphs of cutwidth k can be combined to produce a graph with cutwidth greater than k. The constructions, illustrated in Figure 4, also hold for the VERTEX SEPARATION problem [12]. For VERTEX SEPARATION, if each of the cost k graphs in (a) (resp. (c)) is an obstruction for $(k-1)$-VS, then the resulting graph is an obstruction for k-VS (resp. $(k+1)$-VS) [12]. While combining obstructions for $(k-1)$-MINCUT in one of these ways will not necessarily produce an obstruction for k-MINCUT, the resulting graph does *contain* such an obstruction.

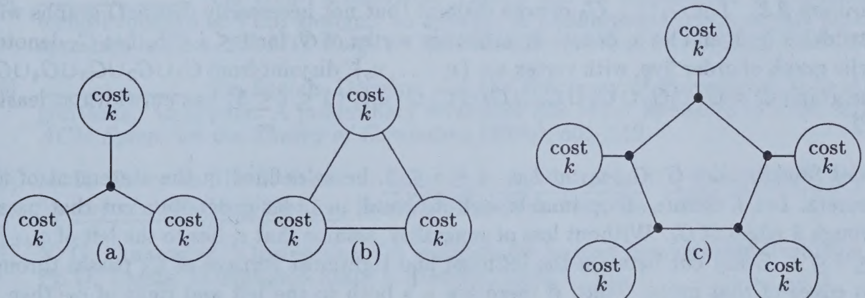


Figure 4: Constructions of Theorems 3.6, 3.7, and 3.8

Theorem 3.6. Let G_1, G_2, and G_3 denote disjoint (but not necessarily distinct) graphs with cutwidth k, let v_i denote an arbitrary vertex of G_i for $1 \leq i \leq 3$, and let v denote an isolated vertex not in $G_1 \cup G_2 \cup G_3$. The graph $G = G_1 \cup G_2 \cup G_3 \cup \{v\} \cup \{vv_i : 1 \leq i \leq 3\}$ has cutwidth at least $k+1$.

Proof. Let G, G_1, G_2, G_3, and v be as defined in the statement of the theorem. Let L denote an optimal layout of G, and, in L, let c_i denote a cut that passes through k edges of G_i. Without loss of generality, assume that c_1 lies to the left of c_2 which lies to the left of c_3. If v_1 (v_3) is to the right (left) of c_2, then c_2 also passes through at least one edge of G_1 (G_3). Therefore, assume v_1 and v_3 are to the left and right, respectively, of cut c_2. If v is to the left of c_2, then edge vv_3 is cut by c_2, while if v is to the right of c_2, then c_2 cuts edge vv_1. In any case, cut c_2 passes through at least $k+1$ edges, and $cw(G) \geq k+1$. □

The construction in Theorem 3.6 is a generalization of a result in [2] where it was shown that the graph constructed by identifying a leaf in each of three smallest degree three trees of cutwidth k produced a smallest degree three tree of cutwidth $k+1$. If the G_i's in Theorem 3.6 are trees, and the v_i's are leaves, G has three vertices of degree two. Reducing G by eliminating the degree two vertices gives the same tree as would be obtained from the construction of [2]. The construction in Theorem 3.6 holds for any graphs and arbitrarily chosen vertices. As graph (b) in Figure 3 illustrates, this construction may produce a graph of cost $k+2$. As graph (d) in Figure 3 shows, the construction in Theorem 3.7 may also produce a graph with cost $k+2$.

Theorem 3.7. Let G_1, G_2, and G_3 denote disjoint (but not necessarily distinct) graphs with cutwidth k, and let u_i and v_i denote arbitrary (but not necessarily distinct) vertices of G_i for $1 \leq i \leq 3$. The graph $G = G_1 \cup G_2 \cup G_3 \cup \{v_1v_2, u_1v_3, u_2u_3\}$ has cutwidth at least $k+1$.

Proof. Let G, G_i, u_i, and v_i, $1 \leq i \leq 3$, be as defined in the statement of the theorem. Let L denote an optimal layout of G, and, in L, let c_i denote a cut that passes through k edges of G_i. Without loss of generality, assume that c_1 lies to the left of c_2 which lies to the left of c_3. If v_1 and v_2 or u_2 and u_3 or u_1 and v_3 are on opposite sides of c_2, then c_2 passes through at least $k+1$ edges. But, if u_1 and v_3 are on the same side of c_2, then the connectivity of G means c_2 passes through at least one edge of G_1 or G_3. Therefore, $cw(G) \geq k+1$. □

Theorem 3.8. Let $G_1, \ldots, G_5$ denote disjoint (but not necessarily distinct) graphs with cutwidth $k \geq 2$, and let u_i denote an arbitrary vertex of G_i for $1 \leq i \leq 5$. Let C_5 denote a cyclic graph of order five, with vertex set $\{v_1, \ldots, v_5\}$, disjoint from $G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5$. The graph $G = G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5 \cup C_5 \cup \{u_iv_i : 1 \leq i \leq 5\}$ has cutwidth at least k + 2.

Proof Sketch. Let G, G_i, u_i, and v_i, $1 \leq i \leq 5$, be as defined in the statement of the theorem. Let L denote an optimal layout of G and, in L, let c_i denote a cut that passes through k edges of G_i. Without loss of generality, assume that c_i lies to the left of c_{i+1} for $1 \leq i \leq 4$. Every cut between the leftmost and rightmost vertices of C_5 passes through two edges of that cycle. Thus, if there are v_i's both to the left and right of c_3, then c_3 passes through at least $k+2$ edges. Suppose then that all of the v_i's are to the right of c_3. Since G is connected, c_3 must pass through both u_1v_1 or an edge incident on a vertex of G_1, and through u_2v_2 or an edge incident on a vertex of G_2. Therefore, c_3 passes through at least $k+2$ edges, and $cw(G) \geq k+2$. □

4. Conclusion

In this paper we have looked ways of obtaining elements of obstruction sets for the fixed parameter variants of two $\mathcal{NP}$-complete linear layout problems: VERTEX SEPARATION and MIN CUT LINEAR ARRANGEMENT. We have looked at both problem specific methods and at general constructions in which obstructions can be combined to produce a graph of higher cost. Interestingly, the constructions in Theorems 3.6–3.8 also hold for the BANDWIDTH [10] problem even though it is not closed under either the minor or immersion orders. Whether these methods can be applied to other layout problems is an open question.

References

[1] R. L. Bryant, M. R. Fellows, N. G. Kinnersley and M. A. Langston, "On Finding Obstruction Sets and Polynomial-Time Algorithms for Gate Matrix Layout," *Proc. 25th Allerton Conf. on Communication, Control, and Computing* (1987), 397–398.

[2] M-J. Chung, F. Makedon, I. H. Sudborough and J. Turner, "Polynomial Time Al-

gorithms for the Min Cut Problem on Degree Restricted Trees," *SIAM J. Comput.* 14 (1985), 158–177.

[3] N. Deo, M. S. Krishnamoorthy and M. A. Langston, "Exact and Approximate Solutions for the Gate Matrix Layout Problem," *IEEE Trans. on Computer-Aided Design* 6 (1987), 79–84.

[4] J. A. Ellis, I. H. Sudborough and J.S. Turner, "Graph Separation and Search Number," Technical Report DCS-66-IR, Department of Computer Science, University of Victoria, August 1987.

[5] M. R. Fellows, "The Robertson-Seymour Theorems: A Survey of Applications," *Contemporary Mathematics* 89 (1989), 1–18.

[6] M. R. Fellows and M. A. Langston, "Nonconstructive Advances in Polynomial-Time Complexity," *Info. Proc. Letters* 26 (1987), 157–162.

[7] ————————, "Nonconstructive Tools for Proving Polynomial–Time Decidability," *J. of the ACM* 35 (1988), 727–739.

[8] ————————, "On Well-Partial-Order Theory and Its Application to Combinatorial Problems of VLSI Design," to appear. A preliminary version of this paper appeared in *Proc. 5th MIT Conf. on Advanced Research in VLSI* (1988), 315–327.

[9] ————————, "On Search, Decision and the Efficiency of Polynomial-Time Algorithms," to appear. A preliminary version of this paper appeared in *Proc. 21st ACM Symp. on the Theory of Computing* (1989), 501–512.

[10] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness, Freeman, San Francisco, CA, 1979.

[11] N. G. Kinnersley, "Obstruction Set Isolation for Layout Permutation Problems," Ph.D. Thesis, Washington State University, 1989.

[12] N. G. Kinnersley and M. A. Langston, "Obstruction Set Isolation for the Gate Matrix Layout Problem," in preparation.

[13] M. Kirousis and C. H. Papadimitriou, "Searching and Pebbling," *Theoretical Computer Science* 47 (1986), 205–218.

[14] T. Lengauer, "Black-White Pebbles and Graph Separation," *Acta Inf.* 16 (1981), 465–475.

[15] F. Makedon and I. H. Sudborough, "On Minimizing Width in Linear Layouts," to appear.

[16] B. Monien and I. H. Sudborough, "Min Cut is $\mathcal{NP}$-Complete for Edge Weighted Trees," *Theor. Comput. Sci.* 58 (1988), 143–154.

[17] N. Robertson and P. D. Seymour, "Graph Minors I. Excluding a Forest," *J. Comb. Th. Ser. B* 35 (1983), 39–61.

[18] ————————, "Graph Minors IV. Tree-Width and Well-Quasi-Ordering," *J. Comb. Th. Ser. B* 48 (1990), 227–254.

[19] ————————, "Graph Minors XIII. The Disjoint Paths Problem," to appear.

[20] ————————, "Graph Minors XVI. Wagner's Conjecture," to appear.

[21] M. Yanakakis, "A Polynomial Algorithm for the Min-Cut Linear Arrangement of Trees," *JACM* 32 (1985), 950–988.

An Efficient Adaptive Algorithm for Distributed Self Diagnosis in Multiprocessing Systems

Yonina Rosen and Robert Gnacek
Department of Electrical Engineering
University of Delaware
Newark, DE 19716
(302) 451-6660

Abstract

This paper introduces a new algorithm for efficient system level fault diagnosis, which is fully distributed, highly reliable, and very efficient with regard to the required tests and therefore testing time. This adaptive algorithm both diagnoses and disables all the faulty units, using only local diagnosers that perform in parallel.

I. Introduction

Modern computer systems, particularly those based on distributed and parallel computing, consist of increasingly large numbers of interconnected processing units. The large number of processors in such systems increases the likelihood of having faulty units in the system, leading to performance degradation and, often, system failure. Thus, a crucial issue in all levels of computer systems is their fault tolerance. This depends on the ability to identify the faulty units, and then repair, replace or disable them. The required high reliability and the inaccessibility of the individual units to the user in many applications make automatic or self fault diagnosis an important goal, which has incited the rapid developments in the theory of system level fault diagnosis in recent years.

The basic model for system level diagnosis, based on the classical digraph model, was developed by Preparata, Metze and Chien [1]. In this model, the system is partitioned into units u_i, each one able to test some subset of the other units. The combined test outcomes are represented as a directed graph, where a unit u_i is depicted as a node in the graph, and the existence of a test of u_j by u_i is depicted as an arc from u_i to u_j. The link (u_i, u_j) has a binary weight $a_{i,j}$, which is 0 if the test result is *pass* (the testing unit u_i evaluates the tested unit u_j to be fault-free), and 1 if the test result is *fail* (the testing unit u_i evaluates the tested unit u_j as faulty). The PMC model presented in [1] assumes that a fault-free unit's evaluation of other units is always correct, while a faulty unit's evaluation of other units is arbitrary. This is referred to as the *symmetric test invalidation* digraph model. The model also assumes that all faults are permanent and that all tests are complete.

The goal of the diagnosis process is to uniquely identify the faulty units in the system from the set of test outcomes $\{a_{i,j}\}$, called the *syndrome*. As shown in [1], a system of n

units can be *one-step t-fault diagnosable* only if the total number of faulty units in the system does not exceed the upper bound t, satisfying $n \geq 2t+1$. Also, in such a system every unit has to be tested by at least t other units. A one-step t-fault diagnosable system is said to be optimal if $n = 2t + 1$ and every unit is tested by exactly t units. A class of such optimal systems are $D_{\delta t}$ designs where $(\delta, n) = 1$ or, equivalently, D_{1t} designs.

The work presented in [1] has given the theoretical basis for most of the recent development in system level fault diagnosis. Subsequently, much research has been performed based on the PMC model. More general necessary and sufficient conditions for t-fault diagnosability were characterized by Hakimi and Amin [2], and later by others. Several generalizations have been proposed with regard to the possible test outcomes, that are more appropriate in some cases, such as the *asymmetric test invalidation* model [3]-[5], the *partial tester* model [6], and the *soft-fail* model [8], that can include intermittent faults [7] and faulty communication links [9]. Another generalization is the probabilistic weighted model [10]-[12].

In all the above work, the diagnosis procedure requires the results of at least nt test results. Furthermore, in most applications these tests cannot all be performed in parallel, which increases the testing time, and the algorithm can only be applied after all the tests have been completed. This makes an on-line implementation of the diagnosis process very difficult.

A smaller number of tests can be achieved by using sequential diagnosis. This, however, suffers from the disadvantages that the total number of faulty units has to be smaller than the upper bound for one-step diagnosis, only one faulty unit can be diagnosed from these tests, and the diagnosis procedure cannot continue until the identified faulty unit is replaced, which may be time consuming and sometimes impossible.

In recent years, a promising approach towards more efficient fault diagnosis has been introduced, where at least some of the testing and diagnosis are performed adaptively. Such an approach was first presented in [13], where the tests are chosen and implemented adaptively until a fault-free unit is identified. Then, this unit is used as a tester to identify all faulty units. Diagnosis algorithms based on this approach were developed in [14] for both the symmetric and asymmetric test invalidation models, under the assumption that every unit can test every other unit. This adaptive diagnosis process requires $n+2t-2$ adaptive tests to diagnose all the faulty units in a system with n units and at most t faults. These results were generalized to probabilistically diagnosable systems in [12]. Two different adaptive diagnosis algorithms were proposed in [15]-[16]. These algorithms also assume that every unit can test all others, and only use the results of $n+t-1$ tests, which is shown to be the lower bound on the number of required tests. However, these algorithms are not applicable to $D_{\delta t}$ designs.

The above methods, both the adaptive and non-adaptive ones, are based on a centralized diagnosis process. They all assume, either explicitly or implicitly, the existence of a central unit that observes all the test results and then performs the specific diagnosis algorithm to identify the faulty units. Some recent works have considered the application of system diagnosis to distributed computing systems [17]-[18]. A distributed approach was proposed in [19], where each unit determines a test table of the system, and then these test tables are compared to diagnose the faulty units. Other distributed diagnosis algorithms were developed for other models, like faulty links and nonhomogeneous systems [20], partial tester [21], and the soft-fail model [8]. Another distributed algorithm [23] uses the model of [24]. This algorithm, too, is based on the "view" of each unit on the fault situation of the system. Like most of these distributed methods, while eliminating the need for a central observer, some central element is still required to implement the diagnosis algorithm. This central controller is still responsible for determining the faulty units, as well as either repairing them or determining some good set of units that can remain in operation. [18] addresses the problem of how the user can identify the faulty units, given the results of the distributed algorithm in [24], using probing units.

A fully distributed algorithm, where the whole diagnosis process is performed by local diagnosers was presented in [25]-[26]. In addition to eliminating the need for any central element, this algorithm also disables all the faulty units. However, the allowable number of faulty units may be smaller than the upper bound for some systems, and the required number of tests may be larger than in one-step diagnosis.

In this paper we propose a new algorithm for system level fault diagnosis, that overcomes many of the above limitations. The proposed algorithm is very reliable since it is able to diagnose all the faulty units in the system where the number of faulty units is up to the upper bound. Furthermore, it does not depend on the reliability of a central diagnoser or controller, as it is fully distributed and does not require the use of such a central element at any stage. Furthermore, it disables all the faulty units in addition to identifying them. The proposed algorithm is efficient in its required number of tests, and therefore testing time, as it uses an adaptive method, where after an initial set of n tests all further testing, as well as diagnosis, is determined adaptively based on the state of the system. The proposed algorithm is presented in the next section.

II. The Adaptive Algorithm

In this section we present a new adaptive and distributed algorithm for system diagnosis. We adhere to the assumptions of the graph-theoretical model in [1]. We assume that each unit has the *ability* to test t other units, so that all *possible* tests form a D_{1t} design (or any $D_{\delta t}$ design where $(\delta, n) = 1$). For any unit u_i, the t units that u_i has the ability to test are said to be in the *connection domain* of u_i. The connection domain of any unit u_i consists of the set of units $\{u_{(i+1)\ mod\ n}, \cdots, u_{(i+t)\ mod\ n}\}$. Note that most of these tests, unlike in one-step diagnosis, will not be required. Each unit is assumed to be enabled unless it has been disabled by the algorithm. A disabled unit continues to do testing but does not perform computations. Each unit contains a local diagnoser that can read all test results of and by this unit, and can disable any unit tested by this unit.

Let us denote the last unit testing u_i as $u_{j(i)}$ and the last unit tested by u_i as $u_{l(i)}$. For each local diagnoser corresponding to u_i, the following steps are performed:

Step 0: Initially, all units are enabled. Units are connected in a single loop configuration (each unit is connected to the next unit in a clockwise direction). Every unit u_i in the system stores the index of the unit testing it (j(i)) and the index of the unit it is testing (l(i)). Initially, $j(i) = (i-1)\ mod\ n$ and $l(i) = (i+1)\ mod\ n$ for all i. This step is only done once.

Step 1: For every unit u_i, if u_i is enabled AND the last test result on u_i is 0 ($a_{j(i),i} = 0$) AND the unit u_i is testing is seen as faulty ($a_{i,l(i)} = 1$), then u_i sends a disabling signal that $u_{l(i)}$ stores.

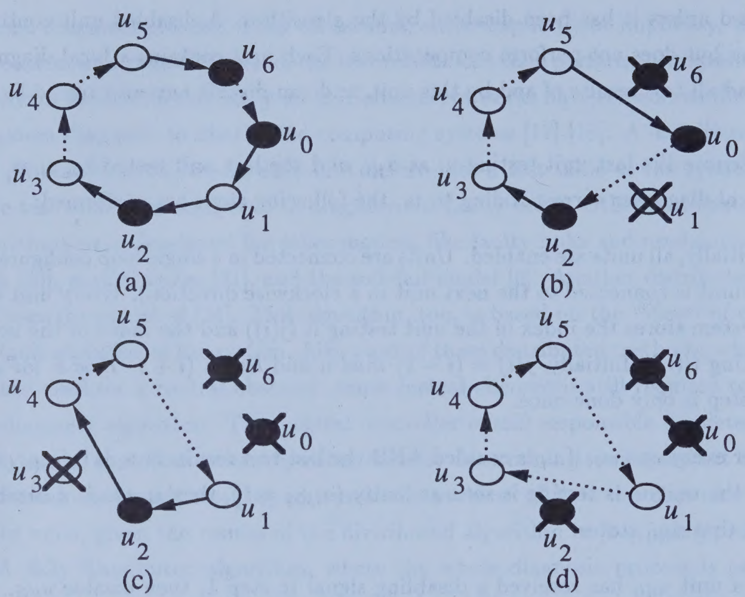
Step 2: If a unit $u_{l(i)}$ has received a disabling signal in step 1, then disable $u_{l(i)}$. $u_{l(i)}$ records that u_i is the unit that disabled it and it also sets its next decision link, $l(l(i)) = (l(i)+1)\ mod\ n$.

Step 3: For any unit disabled in step 2, enable any unit u_m that it may have previously disabled. Set $j(m+1) = m$ for all m.

Step 4: For every unit u_i that has disabled a unit in step 2, if $u_{(l(i)+1)\ mod\ n}$ is in the connection domain of u_i, then a connection between u_i and $u_{(l(i)+1)\ mod\ n}$ is made. The value of $l(i)$ is incremented by one ($l(i) = (l(i)+1)\ mod\ n$) because of this new connection and the link decision $a_{i,l(i)}$ is made. Also, $u_{l(i)}$ sees this decision as its last test result so $j(l(i)) = i$.

The above steps constitute one application of the adaptive algorithm. They are to be applied continually starting at step 1 after the initial application. After a finite number of applications a steady state is reached, where all the faulty units are disabled and all the fault-free units are enabled. Furthermore, all fault-free units are interconnected in the steady state.

A point about the algorithm needs to be made. To minimize the number of tests, memory is necessary in the circuitry that implements the algorithm. Specifically, for every unit, the algorithm circuitry needs to be able to store not only the index of the unit testing and being tested by that unit, but also have the ability to store the result of every test a unit has made along with the index of every unit a unit may disable. If it took 1 memory unit to store each of these, each unit would need $2t+2$ memory units (t memory units to store the index of the units a unit has disabled and t memory units



(a) (b) (c) (d)

Figure 1: An example of the algorithm on a system of where $n = 7$, $t = 3$ (u_0, u_2, u_6). (a) shows the initial conditions of the system and (b), (c), and (d) show the system conditions after one, two, and three applications of the adaptive algorithm respectively. Subsequent applications of the algorithm will not change the enable/disable status of the units.

to store the test results on these units along with 2 memory units to store $l(i)$ and $j(i)$). By allowing the retesting of units, this memory requirement can be avoided but further testing reduces the efficiency of the algorithm.

III. Simulation Results

The performance of the algorithm is demonstrated by the following two examples. In both examples, we consider a system with 7 units, 3 of which are faulty. A worst case scenario where all the test results made by faulty units are incorrect is assumed. Faulty units are shown as black circles while fault-free units are white. A unit is enabled unless it has an X over it, in which case it is disabled. A *fail* test result is shown as a solid arc, while a *pass* test result is shown as a dotted arc.

Example 1

Fig. 1a shows the initial setup of this system. At this point, all the units are connected in a single loop configuration and they are all enabled. After one application of the algorithm, u_5 receives a "pass" test result from u_4 and u_6 "fails" its test from u_5 so u_5 disables u_6. Because u_6 and u_0 are faulty, u_1 is also disabled. All other units remain enabled. After disabling these two units, u_0 and u_5 make testing links to u_2 and u_0

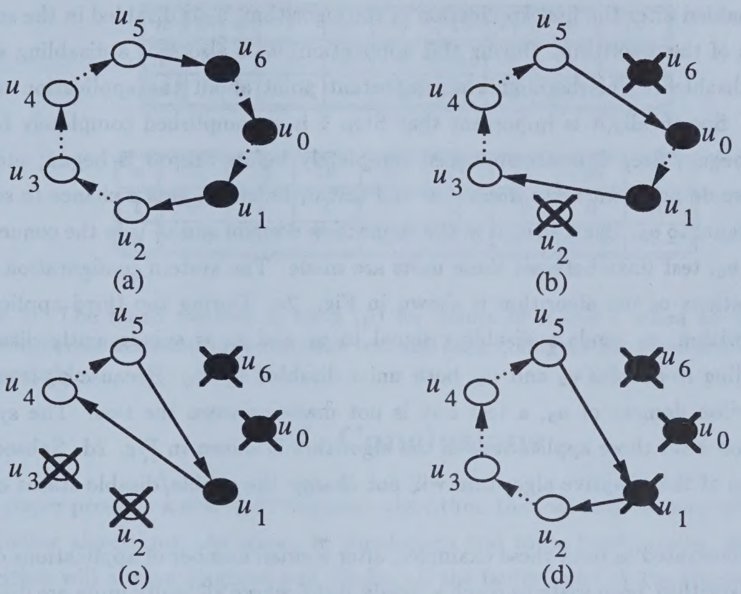


Figure 2: An example of the algorithm on a system of where $n = 7$, $t = 3$ (u_0, u_1, u_6). (a) shows the initial conditions of the system and (b), (c), and (d) show the system conditions after one, two, and three applications of the adaptive algorithm respectively. Subsequent applications of the algorithm will not change the enable/disable status of the units.

respectively (both are in each units connection domain). The system configuration after 1 application is shown in fig. 1b. Fig. 1c shows the system after a second application of the adaptive algorithm. At this point, u_0 is disabled by u_5 and consequently, u_1 is enabled (see step 3). The unit u_3 is disabled by u_2 because of the incorrect test results of u_0 and u_2. u_6 remains disabled (it would take u_5 being disabled to enable it) and all the other units remain enabled. After disabling u_3, u_2 makes a testing link to u_4. After disabling u_0, u_5 makes a testing link to u_1. At the next application of the adaptive algorithm, u_1 disables u_2. The disabling of u_2 triggers the re-enabling of u_3. The system configuration after three applications is shown in fig. 1d. At this point, subsequent applications of the adaptive algorithm to the system will not change the enable/disable status of the units in the system.

Example 2

Again, we look at a system where $n = 7$ and $t = 3$. In this system, as shown in Fig. 2a, all three faulty units are found consecutively in the system. After one application of the algorithm, u_5 disables u_6 and because both u_0 and u_1 are faulty, u_2 is disabled. All other units remain enabled. After disabling u_6 and u_2, u_5 and u_1 make test links to u_0 and u_3 respectively. This is shown in Fig. 2b. Again, since u_0 and u_1 are faulty and still enabled after the first application of the algorithm, u_3 is disabled in the second application of the algorithm. During this application, u_0 is also sent a disabling signal and then disabled. This highlights an important point about the application of the algorithm. Specifically,it is important that Step 1 is accomplished completely before Step 2 is begun, Step 2 is accomplished completely before Step 3 is begun, etc. In this case, we do not want u_5 to disable u_0 and test u_1 before u_1 gets a chance to send a disabling signal to u_3. Because u_4 is in the connection domain and u_1 is in the connection domain of u_5, test links between these units are made. The system configuration after two applications of the algorithm is shown in Fig. 2c. During the third application of the algorithm, u_5 sends a disabling signal to u_1 and u_1 is subsequently disabled. This disabling re-enables u_2 and u_3, both units disabled by u_1. Because u_2 is not in the connection domain of u_5, a test link is not made between the two. The system configuration after three applications of the algorithm is shown in Fig. 2d. Subsequent applications of the adaptive algorithm will not change the enable/disable status of the system.

As demonstrated in both these examples, after a small number of applications of the adaptive algorithm, both systems reach a steady state, where all faulty units are disabled and all fault-free units are enabled and interconnected. Even when all the faulty units are concentrated consecutively in a system as in Example 2, as long as $n \geq 2t + 1$, the system will be corrected. Also, even though we assumed the worst case with regard to test results from faulty units, the total number of tests required for each example, 12 and 11 for Example 1 and Example 2 respectively, is much smaller than that required in one-step diagnosis (21).

Discussion

To verify the operation of the adaptive algorithm, a simulation program was written that demonstrated the performance of the proposed algorithm. The program determined every possible configuration of units given n and t and then ran the adaptive algorithm on them. It was found that for any system where $n \geq 2t+1$ and every unit had t units in its connection domain, after a finite number of applications of the algorithm, all faulty units were disabled and all fault-free units were enabled and interconnected.

The adaptive algorithm was found to be efficient with regard to the number of tests and therefore testing and diagnosis time. Table 1 gives the mean number of tests, denoted μ, for varying values of n and t when all faulty units make incorrect test decisions and each configuration for varying n and t is equiprobable. Two important points need to be made. First of all, there seems to be a linear relationship between t and μ. This linear relationship suggests that very large systems can be accommodated. Secondly, the number of tests required is much smaller than one-step diagnosis (nt) and becomes more significant for larger systems.

n	5	6	7	8	9	10	11	12	13
t	2	2	3	3	4	4	5	5	6
μ	7	8	11.2	12	15.6	16	20	20.3	24.3

n	14	15	16	17	18	19	20	21
t	6	7	7	8	8	9	9	10
μ	24.5	28.6	28.9	32.9	33.2	37.1	37.6	41.2

Table 1: The mean number of tests (μ) for values of n and t when all faulty units make incorrect decisions on units they test and each configuration for a given n and t is equiprobable.

IV. Conclusions

This paper presents a new fault diagnosis algorithm, that overcomes many of limitations of existing algorithms. As shown in simulations and to be later proven, the proposed algorithm will always diagnose and disable all the faulty units in the system up to the upper bound given in [1], and is thus highly reliable. It is fully distributed and does not require a central element at any stage , which further increases reliability. Local diagnosers are only required to get information from local neighbors and perform the algorithm in parallel. This algorithm inherently includes not only the diagnosis but also the disabling of faulty units and the reconfiguration of links between units, so that all the faulty units are disabled and all the fault-free units are enabled and interconnected. The faulty units can then be repaired and will automatically be added to the enabled units, or can be left disabled for graceful degradation. The algorithm is efficient with regard to the required number of tests and therefore the testing and diagnosis time. It requires a much smaller number of tests than in one-step fault diagnosis, even when the number of faulty units achieves the maximal number of t. Furthermore, the number of tests is proportional to the number of units that are actually faulty. This is important since in many situations no units are faulty, in which case only n tests will be performed in one step. Then, if some units are faulty, more tests will be added, only as needed up to the worst case where t units are faulty. This is achieved by using an adaptive method, where both the diagnosis and disabling of units and the activation of test links are dynamic operations that are based on the last set of test results.

References

[1] F.P. Preparata, G. Metze, R.T. Chien, “On the Connection Assignment Problem of Diagnosable Systems”,*IEEE Transactions on Electronic Computers*, vol. EC-16, Dec.

1967, pp. 848-854.

[2] S.L.Hakimi, A.T.Amin, "Characterization of Connection Assignment of Diagnosable Systems", *IEEE Transactions on Computers*, vol. C-23, Jan. 1974, pp. 86-88.

[3] F. Barsi, G. Grandoni, P. Maestrini, "A Theory of Diagnosability of Digital Systems", *IEEE Transactions on Computers*, vol. C-25, No. 6, pp. 585-593, June 1976.

[4] G.G.L. Meyer,"A Diagnosis Algorithm for the BGM System Level Fault Model", *IEEE Transactions on Computers*, vol. C-33, pp. 756-758, Aug. 1984.

[5] J. Narasimhan, K. Nakajima, "An Algorithm for Determining the Fault Diagnosability of a System", *IEEE Transactions on Computers*, vol. C-35, pp. 1004- 1008, Nov. 1986.

[6] L. Simoncini, A.D. Friedman, "Incomplete Fault Coverage in Modular Multiprocessor Systems", *Proceedings ACM Annual Conference*, Washington D.C., pp. 210- 216, Dec. 1978.

[7] S. Mallela, G. M. Masson, "Diagnosable Systems for Intermittent Faults", *IEEE Transactions on Computers*, vol. C-27, No. 6, pp.560-566, June 1978.

[8] C. Yang, G.M. Masson, "A Distributed Algorithm for Fault Diagnosis in Systems with Soft Failures", *IEEE Transactions on Computers*, vol. C-37, Nov. 1988, pp. 1476-1480.

[9] C. Yang, G.M. Masson, "Hybrid Fault Diagnosability with Unreliable Communication Links", *IEEE Transactions on Computers*, vol. 37, No. 2, pp. 175- 181, Feb. 1988.

[10] S.N. Maheshwari, S.L. Hakimi, "On Models for Diagnosable Systems and Probabilistic Fault Diagnosis", *IEEE Transactions on Computers*, vol. C-25, No. 3, pp. 228-236, March 1976.

[11] A.T. Dahbura, "An Efficient Algorithm for Identifying the Most-Likely Fault Set in a Probabilistically Diagnosable System", *IEEE Transactions on Computers*, vol. C-36, pp. 354-356, April 1986.

[12] F.W. Shih, K. Nakajima, "Adaptive Diagnosis for Probabilistically Diagnosable Systems", *Proceedings of the 26th Allerton Conference on Communication, Control and Computing*, Allerton, Illinois, September 1988.

[13] K. Nakajima, "A New Approach to System Diagnosis", *Proceedings of the 19th Allerton Conference on Communication, Control and Computing*, pp. 697- 706, Sept. 1981.

[14] S.L. Hakimi, K. Nakajima, "On Adaptive System Diagnosis", *IEEE Tranactions on Computers*, vol. C-33, No. 3, March 1984.

[15] P.Y. Wu, "Partial Solution to Problem 81-6", *J. Algorithms*, vol.3, pp. 379-380, 1982.

[16] P.M. Blecher, "On a Logical Problem", Discrete Math., vol. 43, pp. 107-110, 1983.

[17] S.H. Hosseini, J.G. Kuhl, S.M. Reddy, "A Diagnosis Algorithm for Distributed Computing Systems with Dynamic Failure and Repair", *IEEE Transactions on Computers*, vol. C-33, No. 3, pp. 223-233, March 1984.

[18] S.E. Kreutzer, S.L. Hakimi, "Distributed Diagnosis and the System User", *IEEE Transactions on Computers*, vol. C-37, Jan. 1988, pp. 71-79.

[19] G. Meyer, G.M. Masson. "An Efficient Fault Diagnosis Algorithm for Symetric Multiple Processor Architectures", *IEEE Transactions on Computers*, vol. C-27, Nov. 1978, pp. 1059-1063.

[20] S.H. Hosseini, J.G. Kuhl, S.M. Reddy, "On Self-Fault Diagnosis of the Distributed Systems", *IEEE Transactions on Computers*, vol. C-7, Feb. 1988, pp. 248-251.

[21] H. Krawczyk, W.E. Kozlowski, "On the Diagnosability of Multicomputer Systems with Homogeneous and Incomplete Tests", *IEEE Transactions on Computers*, vol. C-37, Nov. 1988, pp. 1419-1421.

[22] C.S. Holt, J.E. Smith, "Self-Diagnosis in Distributed Systems", *IEEE Transactions on Computers*, vol. C-34, Jan. 1985, pp. 19-32.

[23] J.D. Russell, C.R. Kime, "System Fault Diagnosis: Closure and Diagnosability with Repair", *IEEE Transactions on Computers*, vol. C-24, Dec. 1975, pp. 1078 - 1088.

[24] J. Kuhl, S. Reddy, "Distributed Fault Tolerance for Large Multiprocessor Systems", *Proceedings of the 7th International Symposium on Computer Architecture*, June 1981, pp. 23-30.

[25] J.T. Butler, Y. Rosen, "Properties of c-Correctability in Self-Diagnosing Systems, *Proceedings of the 26th Annual Allerton Conference on Communication, Control, and Computing*, Sept. 1988, pp. 408-416.

[26] J.T. Butler, Y. Rosen, " A Characterization of Self-Correcting Systems", Submitted to *IEEE Transactions on Computers*, Sept. 1989.

Reliable Gossip Schemes with Random Link Failures

Krzysztof Diks
Instytut Informatyki
Uniwersytet Warszawski
PKIN, p. 850
00-901 Warszawa, Poland

Andrzej Pelc
Département d'Informatique
Université du Québec à Hull
C.P. 1250, succ. "B"
Hull, Québec J8X 3X7, Canada

Abstract.

Each of n people has a piece of information which should be communicated to everybody else. This is to be done by executing a sequence of two-party telephone calls in which the two participants exchange all information they currently have, in a unit of time. Two parameters to be minimized are: the total number of calls or the total time used. We study the following variation of this well-known gossip problem: assuming that telephone lines fail independently with constant probability $0 < p < 1$, find efficient schemes of calls which assure complete communication with probability converging to 1 as n grows. We show a non-adaptive scheme using O(nlogn) calls and prove that it is asymptotically optimal. We also show an adaptive scheme whose expected number of calls is O(n). Moreover, two time-efficient schemes are proposed: a non-adaptive one using $O(\log^2 n)$ time units and an adaptive one working in expected time O(logn).

This research was supported in part by the second author's grant OGP 0008136 from the Natural Sciences and Engineering Research Council of Canada, and was done during the first author's visit at the Université du Québec à Hull.

1. Introduction.

Each of n people has a piece of information unknown to others which must be communicated to everybody else. This is to be accomplished by a series of two-party telephone calls satisfying the following assumptions:

1. each call requires one unit of time;
2. everybody can participate in at most one call per time unit;
3. during a call the two participants exchange all their current information.

Two important parameters of such a series of calls are: the total number of calls and the total time used. The problem of finding sequences of calls, referred to as **gossip schemes**, which minimize the above parameters has been extensively studied in literature under the name of gossip problem (cf. e.g. [1,2,4,6,7,8,10,11]. For n people the minimum number of calls is 2n-4 (cf. [1,4,6,10]) and the minimum time is $\lceil \log n \rceil$ for n even and $\lceil \log n \rceil + 1$ for n odd. (cf. [2,7,8]). (Unless otherwise specified logarithms are taken with base 2).

The problem of constructing gossip schemes optimal in the above sense becomes more complicated if we assume that some calls may fail and no information is exchanged during a failed call. In [3,5] bounds on the minimum total number of calls were obtained assuming that up to k calls may fail. The results imply that $\Theta(nk)$ calls are sufficient and necessary. It follows from [9] that the minimum time required by a gossip scheme is at least $\lceil \log n \rceil + k$, under the same assumptions.

In this paper we study optimal gossip schemes under assumption that every pair of communicating people is connected by exactly one telephone line and these lines fail independently with fixed probability $0 < p < 1$. No information can be exchanged during a call which uses a faulty line, faults are permanent (the fault status of a line does not change during the scheme) and both parties attempting a call along a faulty line become aware of the fault. Under this scenario we cannot expect to complete communication with absolute certainty and thus we look for highly reliable schemes. A gossip scheme for n people is called **almost safe** if the probability of complete information exchange (everybody learns correctly information held by everybody else) converges to 1 as $n \to \infty$.

In the presence of faulty lines two ways of constructing a gossip scheme are possible. One way is **non-adaptive**, that is all calls have to be predetermined (who and when communicates) before the scheme is started, without the possibility of modifying the sequence of calls depending on which calls succeeded and which failed. Only this approach has been adopted in literature [3,5,9]. (In [5] it was called static). Another way is **adaptive**, that is everybody can decide whom to call in a given time unit depending on the outcome of previous calls. However, in making this decision, one can only take advantage of the information he currently has, that is we do not assume the existence of any central monitor supervising the execution of the scheme.

We study both adaptive and non-adaptive almost safe gossip schemes. An adaptive/non-adaptive almost safe scheme is called **asymptotically optimal** (with respect to the number of calls or of time units used) if it uses $O(f(n))$ calls/time units and every other adaptive/non-adaptive almost safe scheme uses $\Omega(f(n))$ calls/time units. In the non-adaptive version we show a scheme using a total of $O(n\log n)$ calls and taking $O(\log^2 n)$ time units. We prove that it is asymptotically optimal with respect to the number of calls. In the adaptive version the total number of calls and the total time needed are random variables, thus the expected values of these parameters are reasonable measures of gossip scheme performance. For this variant of the problem we present an asymptotically optimal almost safe scheme for which the expected number of calls is $O(n)$ and which works in expected time $O(\log n)$. All our results hold for arbitrary link failure probability $0 < p < 1$.

An elementary part of a gossip scheme is an individual call with specified communicating parties x,y and specified time unit counted since the beginning of the scheme. We say that in time unit i a call is placed between x and y or that x calls y. The first expression can always be put in the more precise second form by fixing an arbitrary order of communicating people and supposing that if we say "a call is placed between x and y", it always means that the person with lower number calls.

2. Non-adaptive almost safe gossiping.

In this section we construct a non-adaptive almost safe gossip scheme which uses few calls and can be completed in short time. A non-adaptive gossip scheme can be modeled by a multigraph (G,E) whose vertices represent communicating people and edges represent calls. An edge is called faulty if it represents a call along a faulty line. Each edge has a natural value t indicating at which unit of time the call should be made. Adjacent edges must have different values (property 2 from the Introduction) and, in order to complete communication, between any (ordered) pair of distinct vertices (u,v) there must be a path $(e_1,\ldots,e_n)$ consisting of non-faulty edges with ascending values.

The following result gives a non-adaptive almost safe gossip scheme using an asymptotically minimum number of calls.

Theorem 1

a) There exists a non-adaptive almost safe gossip scheme using $O(n\log n)$ calls and $O(\log^2 n)$ time units.

b) For any constant $c < -1/(4\log p)$ and any non-adaptive gossip scheme using at most $cn\log n$ calls, the probability of complete information exchange converges to 0, as $n \to \infty$.

Proof.

a) Let c be a constant $> -1/\log(1-(1-p)^2)$ and $k(n) = \lceil c\log n \rceil$. Let $m = \lceil \log(n+1) \rceil$ and consider n so large that $k(n) < 2^{m-3}$. Arrange all communicating people into a binary tree with $m-1$ full levels and the last, m-th level filled from the left side until all people are exhausted. Thus the first level has one element - the root and the last level may be incomplete. We will use the notions of the left son of x ($l(x)$), the right son of x ($r(x)$) and the parent of x ($p(x)$) in the usual way.

A basic module of our scheme involves three people: u, v, w and is executed in two consecutive time units; first a call is placed between u and v and next between v and w. We say that u transmits to w via v.

Our scheme is divided in two major parts: first communicating information from everybody to the root and then broadcasting from the root to everybody. The first part consists of $m-1$ consecutive phases, each taking $8k(n)$ time units. The aim of the i-th phase is to communicate information by all people of level $m-i+1$ to their parents. For $2 \le i \le m-1$, let $a_{i,1}, b_{i,1}, a_{i,2}, b_{i,2}, \ldots, a_{i,j_i}, b_{i,j_i}$, for $j_i = 2^{i-2}$, be the enumeration of the i-th level from left to right. For the sake of uniformity we put $a_{11} = b_{11}$ = the root, on level 1. Here is the description of the $(m-i+1)$-th phase for $m \ge i \ge 2$ (sending information from level i to $i-1$).

For $r = 0,1,\ldots,k(n)-1$ we indicate calls to be placed in time units $8r+1$ to $8r+8$. (# denotes addition mod 2^{m-3}). Let $t = 1,2,\ldots,j_{i-1}$.

time unit	$8r+1$	-	a call between	$l(a_{i-1,t})$	and	$b_{m-1,t\#r}$
	$8r+2$	-		$b_{m-1,t\#r}$	and	$a_{i-1,t}$
	$8r+3$	-		$r(a_{i-1,t})$	and	$b_{m-1,t\#r}$
	$8r+4$	-		$b_{m-1,t\#r}$	and	$a_{i-1,t}$
	$8r+5$	-		$l(b_{i-1,t})$	and	$a_{m-1,t\#r}$
	$8r+6$	-		$a_{m-1,t\#r}$	and	$b_{i-1,t}$
	$8r+7$	-		$r(b_{i-1,t})$	and	$a_{m-1,t\#r}$
	$8r+8$	-		$a_{m-1,t\#r}$	and	$b_{i-1,t}$

Thus in each phase elements of a given level receive information transmitted by both sons via a person from level $m-1$. This level is chosen because it contains sufficiently many people to transmit messages for every value of i. It follows from the above description and from the condition $k(n) < 2^{m-3}$ that at any moment of the scheme every person is engaged in at most one conversation.

The second major part - broadcasting from the root - is a mirror image in time of the first part: the first part was executed in time units 1 to $8k(n)(m-1)$ the second is executed in time units $8k(n)(m-1)+1$ to $16k(n)(m-1)$. Calls placed in time unit i during the first part are exactly repeated in time unit $16k(n)(m-1)-i+1$ during the second part. Since at any time unit of the first part every person talks to at most one other person, the same is true for the second part.

The entire scheme takes $16k(n)(m-1)$ time units. Since $k(n) \in O(\log n)$ and $m \in O(\log n)$, the whole scheme works in time $O(\log^2 n)$. It should be noted that apart from communicating from level m to level m-1, in all other cases it is possible to use twice less time than we do. However we chose to describe a uniform algorithm for all levels. In phase m-i+1 every person from level i transmits to his parent k(n) times via different people. Thus the total number of calls placed in the first part is $2(n-1)k(n)$. The entire scheme uses $4(n-1)k(n) \in O(n\log n)$ calls.

It remains to show that the scheme is almost safe. Consider any pair (u, p(u)). In the multigraph corresponding to the first part of the scheme between every such pair there are k(n) edge-disjoint paths containing 2 edges with ascending values. If for every pair (u,p(u)) at least one of those paths does not contain faulty edges, the scheme assures complete communication: during its first part the root gets information from everybody else and during the second part it broadcasts this information to everybody else. For a single path containing 2 edges the probability that it contains a faulty edge is $q = 1-(1-p)^2$. Thus the probability that for some pair (u,p(u)) each of these k(n) paths contains a faulty edge is at most

$$(n-1)q^{k(n)} \leq (n-1)q^{c\log n} = (n-1)n^{c\log q} = (n-1)n^{c\log(1-(1-p)^2)}$$

which converges to 0 because $c > -1/\log(1-(1-p)^2)$.

b) It is enough to prove that, with probability converging to 1, there exists a vertex x of the corresponding multigraph such that all edges incident to x are faulty. The person represented by x has to guess information held by everybody else, and guesses about different people's information are independent; hence the probability that all guesses are correct converges to 0.

Assume that there are at most $cn\log n$ edges in the multigraph corresponding to a gossip scheme. Let R be the set of vertices of degree at most $4c\log n$. Hence R has size at least $n/2$ and for sufficiently large n there is a set S of mutually non-adjacent vertices of size at least

$n/(9c\log n)$. For $x \in S$ let E_x be the event that at least one edge incident to x is not faulty. Events E_x are independent. Let E be the intersection of all events E_x. We have

$$\mathrm{Prob}(E_x) \leq 1-p^{4c\log n} = 1-n^{4c\log p}$$

and

$$\mathrm{Prob}(E) \leq (1-n^{4c\log p})^{n/(9c\log n)}$$

If $c < -1/(4\log p)$, we have $4c\log p > -1$ and hence

$$(1-n^{4c\log p})^{n/(9c\log n)} \to 0$$

■

3. Adaptive almost safe gossiping.

In this section we show that the adaptive way of constructing almost safe gossip schemes may increase their efficiency on average. In fact we present a scheme for which the average number of calls and the average number of time units used are of the same orders of magnitude as for respective optimal schemes working in a fault-free environment.

Theorem 2.
There exists an adaptive almost safe gossip scheme for which the expected number of calls is $O(n)$ and which works in expected time $O(\log n)$.

Proof.
We arrange all communicating people in a binary tree as in the proof of Theorem 1 and keep the same notation. In particular c, k(n), m, l(x), r(x) and p(x) have identical meaning as before. We also enumerate levels in the same way. Again our scheme consists of two major parts: transmitting information from everybody to the root and broadcasting from the root to everybody.

The basic block of the first part involves three people u,v,w and takes three time units. First a call is placed between u and v, then between v and w and finally again between u and v. We enumerate consecutive periods $p_1, p_2, \ldots$, each taking 3 time units counting since the beginning of the scheme. If the above three calls are placed in a given period p_i we say that u transmits to w via v in p_i. It is clear that after these calls are completed each of u,v,w knows if all were successful. In this case we say that transmission from u to w via v was successful.

The following functions indicate via which person from level m-1 a left/right son of a person x has to transmit information to x if he is supposed to do so in a given period p_i. If x is at level m-1 we define

$$M_l(x,i) = \begin{cases} b_{m-1,j\#t}, & \text{if } x = a_{m-1,j}, \quad i = 4t+1,\ t = 0,1,\ldots \\ a_{m-1,j\#t}, & \text{if } x = b_{m-1,j}, \quad i = 4t+2,\ t = 0,1,\ldots \\ 0, \text{ otherwise} & \text{(no transmission)} \end{cases}$$

$$M_r(x,i) = \begin{cases} b_{m-1,j\#t}, & \text{if } \quad x = a_{m-1,j}, \quad i = 4t+3,\ t = 0,1,\ldots \\ a_{m-1,j\#t}, & \text{if } \quad x = b_{m-1,j}, \quad i = 4t+4,\ t = 0,1,\ldots \\ 0, \text{ otherwise} & \text{(no transmission)} \end{cases}$$

As before # denotes addition modulo 2^{m-3}. Thus any a_j can receive information from his left son only in periods p_{4t+1} and from his right son in periods p_{4t+3}. Similarly, any b_j can receive information from his left son only in periods p_{4t+2} and from his right son only in periods p_{4t+4}.

If x is at level < m-1 we denote by L(x) the unique person y at level m-1 such that $y = l(l(\ldots l(x)\ldots))$. In this case we define $M_l(x,i) = M_l(L(x),i)$ and $M_r(x,i) = M_r(L(x),i)$. Thus the left/right son of a person x at level < m-1 can transmit to x at the same periods that are reserved for analogous transmission in the case of L(x) and the transmission is done via the same person as in the case of L(x).

We are now ready to describe the first part of our scheme. Consider any person y at some level > 1. Let x = p(y). Suppose that y obtained information from one of his sons in period $p_{T'}$ and from the other in period $p_{T''}$ (for people at level m this hypothesis is void and should be neglected). Let T be the least integer greater than max (T', T'') and of the form:

4t +1, if y=l(x) and x=l(p(x))
4t+2, if y=l(x) and x=r(p(x))
4t+3, if y=r(x) and x=l(p(x))
4t+4, if y=r(x) and x=r(p(x))

(For x on level 1 the second condition should be omitted in each case and the odd value taken). In periods p_{T+4h} h = 0,1,... y communicates with x via $M_l(x,T+4h)$ in case y=l(x) and via $M_r(x,T+4h)$ in case y=r(x), until h=k(n) or until transmission from y to x via the indicated person is successful, whichever comes first.

This concludes the description of the first part of our scheme. It is obvious from the construction that different people do not transmit to their parents via the same person in the same period hence everybody talks to at most one person in any time unit.

The second part - broadcasting from the root - is again a sort of mirror image in time of the first part, though now only successful calls are repeated. We shall indicate at which period p_i (depending on the history in the first part of the scheme) a given person broadcasts to his son. Broadcasting is similar to transmitting via v in the first part of the scheme, but simpler. If s transmitted to p(s) via v in some period during the first part and transmission was successful then in the second part the following calls are placed in a corresponding period: no calls involving these three people in the first unit, a call between p(s) and v in the second unit and a call between v and s in the third unit. Periods are chosen in the following way. We first consider the root r. Suppose that it first obtained information from a son s_1 via a person v_1 in period p_i and then from the other son s_2 via a person v_2 in period p_{i+j}, (the corresponding transmissions being successful). The broadcasting part begins at period p_{i+j+1} and starts by broadcasting from r to s_2 via v_2. Then at period p_{i+2j+1} r broadcasts to s_1 via v_1. Now consider any person x at one of the levels 2,...,m-1. Suppose that during the first part of the scheme x first obtained information from his son s_1 via a person v_1 in period p_i and then from the other son s_2 via a person v_2 in period $p_{i+j'}$. Suppose also that in the first part of the scheme x successfully transmitted information to p(x) via a person v in period $p_{i+j'+j''}$ and that during the second part of the scheme, x obtained information from p(x) via v in some period p_j. Then in period $p_{j+j''}$ x broadcasts information to his son s_2 via v_2 and in period $p_{j+j''+j'}$ x broadcasts information to his son s_1 via v_1.

This completes the description of our gossip scheme. Since in the first part everybody was involved in at most one call at a time, the same is true in the second "mirror" part. For any x and v, the probability that the transmission from x to p(x) via v succeeds is $(1-p)^2$ (both lines x— v and v — p(x) must be fault-free). Thus it is easy to see that the expected number of periods used in the first part of the scheme is O(log n). Since each period lasts 3 time units, the first part of the scheme takes an expected number of O(logn) time units. According to the description the second part takes as much time as the first, hence the entire scheme works in expected time O(logn). In order to achieve successful transmission from x to p(x) an expected number of $3\lceil 1/(1-p)^2\rceil$ calls have to be placed. It takes n-1 such transmissions in the first part of the scheme which thus requires an expected number of $3\lceil 1/(1-p)^2\rceil(n-1)$ calls. The second part requires 2(n-1) calls if the root obtained all information at the end of the first part and no calls otherwise (in the latter unlikely event information exchange is not achieved). Hence the expected number of calls in the entire scheme is O(n).

The proof that our gossip scheme is almost safe is the same as in Theorem 1.

■

4. Conclusion.

We studied almost safe gossip schemes under the assumption that telephone lines connecting communicating people are subject to independent random permanent faults with probability $0 < p < 1$. In the non-adaptive setting we showed a scheme using an asymptotically optimal number of calls and working in $O(\log^2 n)$ time. In the adaptive case we showed a scheme using an asymptotically optimal expected number of calls and working in asymptotically optimal expected time.

Another variant of the gossip problem with random failures results from assuming that line failures are not permanent: a line which is fault-free at some point of execution of the scheme can subsequently fail and, on the other hand, a failure can subsequently disappear. This situation can be modeled by supposing that individual calls, rather than lines themselves, fail independently with a fixed probability $0 < p < 1$. Now even failures of calls placed along the same line are independent. Again the goal is to find almost safe schemes which use few calls and can be completed in short time. It is easy to see that both our results hold in this fault model as well.

Another assumption we made is that every pair of communicating people is connected by a line. This of course is costly and can often be impractical, whence a natural question, what is the least number of lines in a network which can assure an almost safe gossip scheme. A closer look at our schemes shows that we do not use all lines: in fact both proposed schemes use only $O(n\log n)$ lines. The same argument as that proving Theorem 1.b) shows that this asymptotic bound cannot be improved for almost safe gossip schemes. Thus our schemes are asymptotically optimal regarding the cost of the supporting telephone network.

References.

1. R. Baker, R. Shostak, Gossips and telephones, Discrete Mathematics 2 (1972), 191-193.
2. A. Bavelas, Communication patterns in task-oriented groups, J. Acoust. Soc. Amer. 22 (1950), 725-730.
3. K.A. Berman, M. Hawrylycz, Telephone problems with failures, SIAM J. Alg. Disc. Meth. 7 (1986), 13-17.
4. R. Bumby, A problem with telephones, SIAM J. Alg. Disc. Meth. 2 (1981), 13-18.
5. R. Haddad, W. Roy, A.A. Schaffer, On gossiping with faulty telephone lines, SIAM J. Alg. Disc. Meth. 8 (1987), 439-445.
6. A. Hajnal, E.C. Milner, E. Szemeredi, A cure for the telephone disease, Canad. Math. Bull. 15 (1972), 447-450.
7. W. Knodel, New gossips and telephones, Discrete Mathematics 13 (1975), 95.
8. H.G. Landau, The distribution of completion times for random communication in a task-oriented group, Bull. Math. Biophys. 16 (1954), 187-201.
9. A.L. Liestman, Fault-tolerant broadcast graphs, Networks 15 (1985), 159-171.
10. A. Seress, Quick gossiping without duplicate transmissions, Graphs and Combinatorics 2 (1986), 363-383.
11. R. Tidjeman, On a telephone problem, Nieuw. Arch. Wisk 19 (1971), 188-192.

Self-Stabilizing Distributed Systems with Binary Machines

Sukumar Ghosh
Department of Computer Science
The University of Iowa
Iowa City, IA 52242

Abstract

Dijkstra [4] introduced the problem of self-stabilization in distributed systems as an interesting exercise for achieving global convergence using local actions. His model consisted of a set of n finite-state machines (each having k states) connected in the form of a ring, and he presented examples of three self-stabilizing systems: (i) $k \geq n$, (ii) $k = 3$ and $k = 4$. It remained an open problem whether non-trivial self-stabilizing systems can be designed using machines having only two states. This paper demonstrates the feasibility of a self-stabilizing system with only binary machines.

1 Introduction

A distributed system traditionally consists of a set of loosely connected state machines which do not share a global memory. Depending on the connectivity and the propagation delay of the signals, each component machine gets a partial view of the global state.

Based on some predefined global criteria, the global states of such a system can be classified into the categories *legitimate* and *illegitimate*. A *self-stabilizing system* is a distributed system which, when started from an arbitrary (and possibly illegitimate) initial state, is always guaranteed to converge to a legitimate state in a finite number of steps. In many distributed systems, the ability to recover from unforeseen perturbations (which could be viewed as transient failures) without the intervention of any external agency is a highly desirable property, and contributes to the stability of the system. This amply shows the importance of studying self-stabilizing systems.

In [4], Dijkstra first demonstrated the feasibility of a self-stabilizing system. His system consisted of n machines $(0, 1, \ldots, n-1)$ connected in the form of a ring. Each machine is a *k-state* machine which can read (i) its own state, (ii) the state of its left neighbor, and (iii) the state of its right neighbor. Depending on some predicate(s) defined over these states, a machine can update its own state. A machine enjoys a *privilege* when such a predicate is true, and a privileged machine updates its own state, which is called a *move*. Furthermore, when more than one machine enjoy privileges, the choice of the machine which is entitled to make a move is determined arbitrarily by *a central demon*.

In the above framework, Dijkstra [4] discussed the feasibility of self-stabilizing systems in three different cases: (i) $k \geq n$, (ii) $k = 3$, and (iii) $k = 4$. The optimality of the three-state solution was demonstrated in [7] [10]. Solutions on a tree topology was demonstrated by Kruijer [9] with machines having at least four states. In [1], Brown et.al. described a set of self-stabilizing token systems that work without the help of a central demon. Their systems have a useful property called *noninterference* – however the simplest solutions requires at least three states for a machine. Based on these results, it is generally agreed that it is not difficult to design self-stabilizing systems with machines having more than two states. However, it remained an open question whether it is possible to design a non-trivial[1] self-stabilizing system with two-state machines only. This paper shows that it is indeed feasible to design a non-trivial self-stabilizing system with binary machines.

The paper is organized as follows. Section 2 introduces the basic concepts and the notations. Section 3 introduces self-stabilizing systems with two-state machines – it begins with the description of a few special systems, and subsequently it presents a generalized version of it on a special topology, along with its correctness proof. Finally, section 6 contains some concluding remarks.

2 Basic Concepts and Notations

The set of possible global states of a distributed system can be partitioned (according to some predefined postcondition) into two subsets : *legitimate* and *illegitimate*. The following global criteria are to be satisfied for the *legitimate states*:

[LS1] In each legitimate state, one or more privileges must be present.

[LS2] In each legitimate state, each possible move would again bring the system to a legitimate state.

[LS3] Each privilege must be present in at least one legitimate state.

[LS4] For any pair of legitimate states, there must exist a sequence of moves transferring the system from one to the other.

A system is called *self-stabilizing*, if and only if, regardless of the initial state, and regardless of the privilege selected each time for the next move, at least one privilege would always be present, and the system is guaranteed to find itself in the legitimate state after a finite number of moves.

---

[1] A trivial system is one in which every possible global state is a legitimate state.

In all his solutions, Dijkstra considered a specific version of LS1 to characterize a legitimate state – he considered a legitimate state to be one in which *exactly one machine enjoys a privilege.* In this paper, we adopt this characterization of the legitimate states.

The lifecycle of a machine i in a self-stabilizing system can be represented as follows:

```
repeat
        if privilege[1] then move[1] fi;
        if privilege[2] then move[2] fi;
        ... ... ... ... ...
        if privilege[m] then move[m] fi;
forever.
```

We would use the symbol $s[i]$ to represent the state of a machine i. Also, the notation

precondition → *postcondition*

would be used to represent the changes caused by a move.

A *unidirectional protocol* is one in which each machine i computes its privilege from only $(s[i], s[i-1])$[2] (or $(s[i], s[i+1])$, but not both) Limitations of unidirectional protocol are apparent from [4] [6], where it has been shown that the number of states k of each machine has to be equal to at least n, which is the total number of machines in the system. *Bidirectional protocols* allow privileges to be defined in terms of both $(s[i], s[i-1])$ and $(s[i], s[i+1])$, and are considered more powerful, since it is possible to devise solutions with arbitrarily large values of n using machines with limited number of states. To explore the possibility of devising a self-stabilizing system with only 2-state machines, unidirectional protocols are therefore ruled out.

3 Self-stabilizing systems with two-state machines

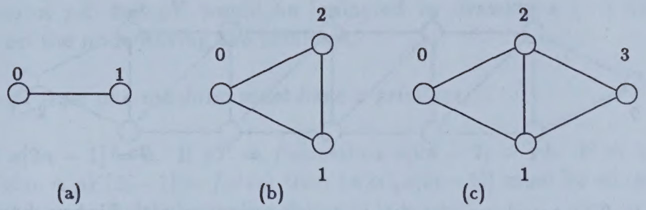
3.1 Some special systems

Let us first demonstrate the existence of self-stabilizing systems with two-state machines on special topologies with $n = 2, 3, 4$[3] Figure 1 shows three special configurations with dissimilar machines which are self-stabilizing. The privileges and the

---

[2] These - and + operations are *mod n* operations, but for the sake of brevity, we omit this detail.

[3] The existence of a self-stabilizing system with three machines each having two states is mentioned in Burns [3].



(a) (b) (c)

Figure 1: Some special self-stabilizing systems with binary machines. The algorithms for self-stabilization are described in Algorithm 1.

moves of these machines for achieving self-stabilization are described in Algorithm 1. We use the symbol b to represent a binary value.

```
Algorithm 1: Self stabilization on special topologies of figure 1.
{For graph 1a}
machine 0: if $(s[0],s[1]) = (b,\neg b)$ then $s[0] := \neg b$ fi;
machine 1: if $(s[0],s[1]) = (b,b)$ then $s[1] := \neg b$ fi;
{For graph 1b}
machine 0: if $(s[0],s[1],s[2]) = (b,b,\neg b)$ then $s[0] := \neg b$ fi;
machine 1: if $(s[0],s[1]) = (b,\neg b)$ then $s[1] := b$ fi;
machine 2: if $(s[1],s[2]) = (b,b)$ then $s[2] := \neg b$ fi;
{For graph 1c}
machine 0: if $(s[0],s[1]) = (b,\neg b)$ then $s[0] := \neg b$ fi;
machine 1: if $(s[1],s[2],s[3]) = (b,b,\neg b)$ then $s[1] := \neg b$ fi;
machine 2: if $(s[0],s[1],s[2]) = (b,b,\neg b)$ then $s[2] := b$ fi;
machine 3: if $(s[2],s[3]) = (b,b)$ then $s[3] := \neg b$ fi;
```

Note: The case $n = 2$ is a trivial one (and therefore, not interesting), since in every possible state of the system is a legitimate state. The cases $n = 3$ and $n = 4$ are of course not trivial. No methodology is given here regarding how these systems were derived. However, the correctness of these can be verified by recording the state transitions with every possible initial state.

3.2 A general system

Our primary target here is to demonstrate the existence of nontrivial self-stabilizing systems with binary machines. This is possible only on topologies which are more complex than a simple ring. The topology that we consider for this purpose is shown in Figure 2.

The system consists of a strongly connected graph of $2n$ nodes which are num-

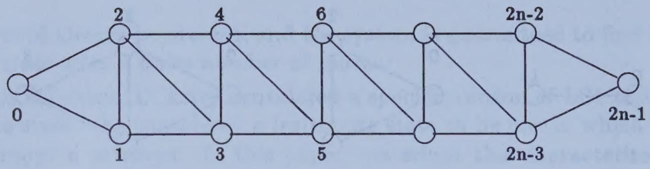


Figure 2: The topology of a self-stabilizing system with binary machines.

bered $(0, 1, \cdots, 2n-1)$. Each machine has only two states: *true* and *false*. In this system, there are two types of *general* machines, called the *x-machines* and the *y-machines*. All the odd numbered machines $(1, 3, \cdots, 2n-3)$ are called *y-machines*, and all the even numbered machines $(2, 4, \cdots, 2n-2)$ are called the *x-machines*. The machines 0 and $(2n-1)$ are *exceptional* machines[4], and following Dijkstra's terminology, we call machine 0 the ***bottom machine*** and machine $(2n-1)$ the *top machine*. A subgraph consisting of the nodes $(2i-1, 2i, 2i+1, 2i+2)$ would be called a *block*.

The algorithm for self-stabilization on the above system is given below. We use the symbol b to represent a binary value *true* or *false*.

```
Algorithm 2: Self stabilization using binary machines.
{For each x-machine 2i}
if($s[2i-2]$,$s[2i-1]$,$s[2i]$,$s[2i+1]$) = $(b,b,\neg b,b)$ then $s[2i]$ := $b$ fi;
{For each y-machine 2i-1}
if ($s[2i-1]$,$s[2i]$,$s[2i+1]$) = $(b,b,\neg b)$ then $s[2i-1]$ := $\neg b$ fi;
{For the bottom machine 0}
if ($s[0]$,$s[1]$) = $(\neg b,b)$ then $s[0]$ := $b$ fi;
{For the top machine 2n-1}
if ($s[2n-1]$,$s[2n-2]$) = $(b,b)$ then $s[2n-1]$:= $\neg b$ fi;
```

3.3 Correctness Criteria

To prove the correctness of the above algorithm, we would use the symbols pX, pY, pT and pB to denote the privileges of the x-machines, the y-machines, the top machine, and the bottom machine respectively. Also, in pictorial representations, the existences of pX and pY would be indicated by drawing a (→) and a (←)[5] respectively on the node having the privilege.

---

[4] The need for an exceptional machine has been explained in [4] [6]. Only in special configurations, Burns et.al. [3] show that the use of an exceptional machine can be avoided.

Lemma 1: *At least one machine must have a privilege.*

Proof: Let $s[2n-1] = b$. If $pT = false$ then $s[2n-2] = \neg b$. If $\forall i \in 1..n-1$, $(pX[2i] = false \wedge pY[2i-1] = false)$ then $(s[2i], s[2i-1])$ must be equal to $(\neg b, b)$ and $s[0]$ must be $\neg b$. But this implies that pB is true.

Thus, in every global state, at least one machine must have a privilege.

□

Lemma 2: $pX[2i] \wedge pY[2i-1] = false$

Proof:

(a) $pX[2i] \Rightarrow s[2i] \neq s[2i-1]$.
(b) $pY[2i-1] \Rightarrow s[2i] = s[2i-1]$.

□

Lemmas 3 – 6 follow directly from Algorithm 2. Therefore, these are presented without proofs.

Lemma 3: *Let* $s[2i] = b$, *where* $b \in \{true, false\}$. *Then*

(a) $pX[2i] \wedge (s[2i+2], s[2i+3] = b, \neg b) \rightarrow \neg pX[2i] \wedge pX[2i+2]$
(b) $pX[2i] \wedge (s[2i+2] = s[2i+3]) \rightarrow \neg pX[2i] \wedge \neg pX[2i+2]$

Note: This lemma shows the rules of propagation of a pX (→). Part (a) corresponds to the propagation of a pX to the right. Part (b) corresponds to the absorption of a pX without generating any new privilege. Note that in both of these cases, the number of privileges never increases.

Lemma 4: *Let* $s[2i+1] = b$, *where* $b \in \{true, false\}$. *Then*

(a) $pY[2i+1] \wedge (s[2i], s[2i-1] = b, b) \rightarrow pY[2i-1] \wedge \neg pY[2i+1]$
(b) $pY[2i+1] \wedge (s[2i], s[2i-1] = b, \neg b) \rightarrow \neg pY[2i-1] \wedge \neg pY[2i+1]$
(c) $pY[2i+1] \wedge (s[2i] = \neg b) \rightarrow \neg pY[2i+1] \wedge pX[2i+2]$

---

[5] This notation has been borrowed from [5].

Note: This lemma shows the rules of propagation of a pY $(\leftarrow)$. Part (a) corresponds to the propagation of a pY to the left. Part (b) corresponds to the absorption of a pY without generating any new privilege. Again, note that in both of these cases, the number of privileges never increases. Part (c) deals with the interesting possibility, when a pY is converted to a pX.

Lemma 5: *Let* $(pX[2i] = true) \wedge (pY[2i+1] = true)$. *If machine* $2i+1$ *moves before machine* $2i$, *then both* $pX[2i]$ *and* $pY[2i+1]$ *are killed and a* $pX[2i+2]$ *is generated. If machine* $2i+1$ *moves after machine* $2i$, *then both* $pX[2i]$ *and* $pY[2i+1]$ *are killed, and a* $pY[2i-1]$ *is generated.*

Note: The essence of this lemma is that if a pX $(\rightarrow)$ propagating from left-to-right meets a pY $(\leftarrow)$ propagating from right-to-left at a *block*, then eventually one of these two privileges is *always* eliminated.

Lemma 6:

(a) if $pX[2n-2] \rightarrow pT$ then $pT \rightarrow pY[2n-3]$
(b) if $pY[1] \rightarrow pB$ then $pB \rightarrow pY[2]$

Note: This lemma shows that the top and the bottom machines essentially act as *reflectors*. The top machine converts a pX $(\rightarrow)$ to a pY $(\leftarrow)$, and the bottom machine converts a pY $(\leftarrow)$ to a pX $(\rightarrow)$.

Based on the above lemmas, the correctness proof can now be constructed.

Theorem 2: *Unless the bottom machine moves, a* pY *can be converted into a* pX *only a finite number of times.*

Proof: Lemma 4(c) shows how a pY may be converted to a pX. Once for some value j $(2 \leq j \leq n-2)$ of i, a $pY[2i+1]$ is converted into a $pX[2i+2]$, this phenomenon cannot repeat unless machine $2i$ also makes a move. But machine $2i$ cannot move, unless machine $2i+1$ moves once more. In summary, the next move by machine $2i+1$ cannot lead to the creation of a $pX[2i+2]$ – it can either lead to the creation of a $pY[2i-1]$ (lemma 4a), or it does not lead to the generation of any new privilege at all (lemma 4b). This implies that if a $pY[2i+1]$ is at all converted to a $pX[2i+2]$ for a second time, then it must occur for some value k of i, such that $k < j$. Therefore, after a finite number of conversions from a $pY[2i+1]$ to a $pX[2i+2]$, the bottom machine has to move at least once.

□

Theorem 2 shows that no y-machine $(2i+1)$ can convert a pY into a pX infinitely often, unless its left neighbor $(2i-1)$ also moves. This can be *abstracted* to the observation that a pY, if not absorbed, *eventually* propagates to the left.

Theorem 3: *If exactly one machine has a privilege, then the system is stabilized.*

Proof: Lemmas 3 – 6 show that the number of privileges never increases. Also, following lemma 1, a privilege cannot be absorbed unless there exists at least one more privilege somewhere in the system. We therefore only demonstrate that the only privilege existing in the system must eventually pass from one machine to another and reach every other machine in a finite number of moves.

When only machine $2i$ has a privilege (pX), condition (a) of lemma 3 holds, so with every move, this privilege propagates one step to the right until it reaches the top node. Part (a) of lemma 6 shows that as the top node moves, the pX is converted to a pY. With no other node having a privilege, condition (a) of lemma 4 holds. So with every move, this privilege propagates one step to the left until it reaches the bottom node. Part (b) of lemma 6 shows that as the bottom machine moves, the pY is converted to a pX. Thus, the one privilege remains the only one and passes from machine to machine. So the system is stabilized.

□

Theorem 4: *Algorithm 2 guarantees self-stabilization.*

Proof: An arbitrary initial state can be portrayed as one in which more than one x-machines and y-machines have privileges (Fig.3). We need to demonstrate that in a finite number of moves, the number of privileges is reduced to one.

As a consequence of lemmas 3 – 6, the number of privileges never increases. A pX, if not absorbed, can propagate only to the right. Similarly, a pY, if not absorbed, *eventually* propagate only to the left. Also, a pX is converted to a pY at the *top* node, and a pY is converted to a pX at the *bottom* node. Under these circumstances, it is inevitable that as long as there are more than one machines enjoying privileges, the condition $pX[2i] \wedge pY[2i+1]$ must eventually hold for some i. According to lemma 6, this always leads to the negation of one of the two privileges. Accordingly, the number of privileges would be reduced in a finite number of moves, until this number drops down to one. According to theorem 3, the system is stabilized at this stage.

□

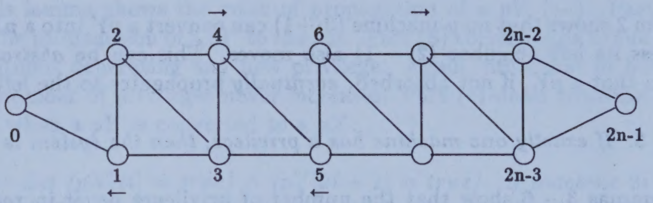


Figure 3: An arbitrary initial state of the self-stabilizing system.

4 Conclusion

This concludes the description and the proof of a self-stabilizing algorithm with the most primitive form of machines which have only two states. The proof of the correctness of Algorithm 2 was based on the assumption that a central demon decides which machine would make a move. The proposed algorithm has however one limitation : it does not work with distributed demons, and the required atomicity is large. This solution may therefore fail where a pair of neighboring machines simultaneously check their privileges and then decide to make moves. An improved algorithm that would work with distributed demons is currently being investigated.

Although not shown in this paper, it is not difficult to demonstrate the impossibility of general self-stabilizing system with two-state machines on a ring topology. The topology of Fig. 2 appears to be the sparsest topology on which self-stabilization is possible with binary machines. The relationship between the topology and the minimum number of states required to achieve self-stabilization still appears to be an open problem.

5 Acknowledgement

Acknowledgements are due to Xiaoming Lin for helpful criticisms, and for uncovering an error in an earlier version of this algorithm.

6 References

1. Brown, G.M., Gouda, M.G. & Wu, C.L. "Token Systems that Self-stabilize" *IEEE Trans. Comput. 38, No. 6, pp. 845 - 852,* June 1989.

2. Burns, J.E. "Self-stabilizing Rings without Demons", *Tech. Report GIT-ICS-87/36, Georgia Institute of Technology,* November 1987.

3. Burns, J.E. & Pachl, J. "Uniform Self-stabilizing Rings", *ACM Trans. Prog. Lang. & Syst. 11, No. 2, pp. 330 - 344*, April 1989.

4. Dijkstra, E.W. "Self-stabilizing Systems in spite of Distributed Control", *Commun. ACM 17, 11, pp. 643-644*, Nov. 1974.

5. Dijkstra, E.W. "A Belated Proof of Self-stabilization", *Distributed Computing 1, No. 1, pp. 5-6*, 1986.

6. Dijkstra, E.W. "Self-stabilization in Spite of Distributed Control", *EWD 391, in Selected Writing on Computers: A Personal Perspective, pp. 41-45*, Springer Verlag, 1982.

7. Ghosh, S. "Understanding Self-stabilization in Distributed Systems : Part 1", *Tech. Report, University of Iowa*, March 1990.

8. Kessels, J.L.W. "An Exercise in Proving Self-stabilization with a Variant Function", *Information Processing Letters 29, No. 2, pp. 39 - 42*, 1988.

9. Kruijer, H.S.M. "Self-stabilization (inspite of distributed control) in Tree-Structured Systems", *Information Processing Letters 8, No. 2, pp. 91 - 95*, 1979.

10. Tchuente, M. "Sur l'auto-stabilisation dans un reseau d'ordinateurs", *RAIRO Informatique Theorique 15, No. 1, pp.47 - 66*, 1981.

Efficient, Dynamically Structured Multiprocess Communication

Yuh-Jzer Joung *joung@sbcs.sunysb.edu*

Scott A. Smolka* *sas@sbcs.sunysb.edu*

Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794-4400

Abstract

We present an efficient distributed algorithm for dynamically structured multiprocess communication in the context of Hybrid CSP [9], where communication primitives are the normal input and output commands of CSP, and the *hybrid* command, which is simply a set of input/output commands to be executed as an atomic unit. The algorithm improves upon an earlier implementation [9] in two significant ways. First, hybrid commands are now allowed to appear as guards, under a *disjointedness* condition that permits an interesting class of multiprocess applications to be expressed. Secondly, the new algorithm is considerably more efficient, using only $O(n)$ messages per guard to establish a multiprocess communication, as opposed to the $O(n^2)$ messages needed by the original algorithm, where n is the number of processes in the system.

1 Introduction

A *multiprocess communication* is a set of I/O actions executed jointly by a number of processes, each of which must be ready to execute its own action for any of the actions in the set to occur. An attempt to participate in a multiprocess communication delays a process until all other participants are available. After the actions are executed, each participating process continues its local computation, usually asynchronously. Languages like CSP [8] and Ada [19] support communication only between two processes. However, for many applications, a higher level of abstraction can be obtained by permitting communication among an *arbitrary* number of processes. For example, consider the well-known dining philosophers problem. The natural unit of process interaction in this setting is between a philosopher and its two neighboring forks; i.e., a multiprocess synchronization involving three processes.

In an earlier paper [9], we made the distinction between two forms of multiprocess communication: *statically structured*, in which processes use global interaction names that uniquely define the participants, and *dynamically structured*, in which a participant specifies only those processes with which it communicates directly. For example, if a process wishes to perform a multicast to several sets of recipients, then in a statically structured scheme each set must be assigned a globally unique interaction name, which appears in both the sender and recipients. A recipient has to specify an interaction name that implicitly defines the other participants. In a dynamic setting, the sender specifies all recipients of a multicast, and each recipient specifies only the sender and *not* the other recipients; no global unique interaction names are required. The result is a more concise and natural specification.

---

*Research supported by the National Science Foundation under Grant CCR-8704309.

Until now, only statically structured schemes have been used in practice, i.e., in actual programming languages supporting multiprocess communication. Examples include *PPSAs* [16], *Action Systems* [1], *Raddle* [6], *LOTOS* [3] and *IP* [7]. In contrast, many algebraic models of concurrency are based upon some notion of dynamically structured multiprocess communication, including *SCCS* [14], *Circal* [13], *CSPS* [18], and *Extended LOTOS* [4]. The primary reason for this dichotomy concerns the tradeoff involving the expressive power of dynamic multiprocess communication versus the implementation efficiency of static multiprocess communication. Below, we discuss this tradeoff further.

In this paper, we consider dynamically structured multiprocess communication within the setting of Hybrid CSP, an extension of Hoare's CSP first presented in [9]. The communication constructs of Hybrid CSP are the normal input and output commands of CSP, and the *hybrid* command, which is simply a bracketed list of input/output commands to be executed as an atomic unit. For example, process p_2 can execute the hybrid command $< p_1?x_2, p_3!x_2 >$ if and only if p_1 is ready to perform a complementary output action *and* p_3 is ready to perform a complementary input action. Execution of p_2's hybrid command may establish a communication among processes other than just p_1, p_2, and p_3, depending on the roles p_2 and p_3 choose to play in the interaction. If, say, p_1 specifies the hybrid command $< p_4?x_1, p_2!x_1 >$, p_3 specifies $< p_2?x_3, p_4!x_3 >$, and, finally, p_4 specifies $< p_3?x_4, p_1!x_4 >$, then the four processes establish a communication in which they circulate the values of x_i's.

Communication commands of Hybrid CSP may be executed nondeterministically as guards of an alternative/repetitive command. As such, the problem of scheduling a multiprocess communication in Hybrid CSP can be cast as a dynamic agreement problem among an arbitrary number of processes on selecting a set of completely compatible communication commands to execute. This problem is in general NP-complete [10]. In order to achieve an efficient distributed implementation of Hybrid CSP, [9] disallows hybrid commands to serve as guards, i.e., only input/output guards are permitted. For some naturally occurring applications of multiprocess communication, this restriction may be too severe.

Consider a system that is configured as in Figure 1, where one of the two controllers r and t from time to time performs a group communication with either the pair of sensors u_1 and u_2, or the pair v_1 and v_2. After the communication, the controller processes the data received from the sensor pair, while the sensors resume collecting new data. The expression for each controller would be

```
*[ $< u_1?x, u_2?y > \longrightarrow process\_data\ (x, y)$
□  $< v_1?x, v_2?y > \longrightarrow process\_data\ (x, y)$ ]
```

and for each sensor would be

```
*[ $r!d \longrightarrow sense\_new\_data\ (d)$
□  $t!d \longrightarrow sense\_new\_data\ (d)$ ]
```

Note that the use of the two hybrid commands as guards in a controller is essential to the correct functioning of the system, since the environment decides whether a controller should communicate with u_1 and u_2, or with v_1 and v_2. Without hybrid guards, an expression of the form

```
*[ $true \longrightarrow < u_1?x, u_2?y >; process\_data\ (x, y)$
□  $true \longrightarrow < v_1?x, v_2?y >; process\_data\ (x, y)$ ]
```

is prone to deadlock, as the decision as to which hybrid command to execute is local to each controller. If a controller, say r, decides to input data from a sensor pair that has died due to a malfunction, then r would be blocked from obtaining data from the other sensor pair.

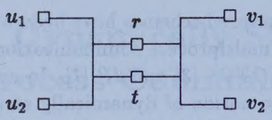


Figure 1: A system with two controllers t and r and two pairs of sensors (u_1, u_2) and (v_1, v_2); the controllers process the data collected by either one of the two pairs.

In this paper we present a distributed algorithm to implement Hybrid CSP communication where hybrid guards are allowed, under the following condition:[1]

> **Disjointedness:** *Let G be the set of communication commands appearing as the guards of an alternative/repetitive command. Then, no two commands in G target the same process for input and no two commands in G target the same process for output.*

For example, the repetitive commands of controllers r and t in the sensors-controllers system satisfy the disjointedness condition as $\{u_1, u_2\} \bigcap \{v_1, v_2\} = \emptyset$. (The disjointedness condition can be weakened further by considering the types of the expressions involved in the communications.) Moreover, our algorithm uses only $O(n)$ messages for each communication guard, as compared to the $O(n^2)$ messages required by the algorithm of [9], where n is the total number of processes in the system.

The structure of the rest of the paper is as follows. Section 2 defines an abstract model of the behavior of Hybrid CSP processes. In Section 3 we present our algorithm under this model. The correctness and message complexity of the algorithm is considered in Section 4. We conclude in Section 5.

2 A Simple Model for Dynamic Multiprocess Synchronization

We consider a distributed system as a set of concurrent processes that communicate by executing hybrid commands. Initially, each process in the system is in its *local computing phase* which does not involve any communication. From time to time, a process enters its *communication phase* where it is ready to execute any one of a set of hybrid commands.

In this paper we address only the synchronization part of a multiprocess communication. As such, a *hybrid* command can be regarded as a set of process names, meaning that the process executing the command wishes to communicate with these named processes. According to the disjointedness condition introduced in Section 1, we require that the set of hybrid commands for which a process is ready to execute in its communication phase be *disjoint*, *i.e.*, the intersection of any two commands in the set is empty.

Let $c_1, \ldots, c_m$ be hybrid commands of processes $p_1, \ldots, p_m$, respectively. These commands are *completely compatible* if c_i names p_j iff c_j names p_i, $1 \leq i \neq j \leq m$, *i.e.*,

$$p_j \in c_i \Longleftrightarrow p_i \in c_j$$

For example, the following three commands of p_1, p_2, and p_3, respectively, are completely compatible:

$$c_1 = < p_2, p_3 >, \quad c_2 = < p_1 >, \quad c_3 = < p_1 >$$

---

[1] This restriction applies naturally to CSP where only input and output guards are permitted. Moreover, under this restriction input and output commands in Hybrid CSP can be regarded as hybrid commands that involve only one action. (In [9] input and output commands are distinguished from hybrid commands because hybrid commands cannot act as guards.)

while the following three commands are not:

$$c_1 = < p_2, p_3 >, \quad c_2 = < p_1 >, \quad c_3 = < p_1, p_2 >$$

We say that there is a *potential interaction* among processes $p_1, \ldots, p_m$ if together they have a set of completely compatible hybrid commands that they are ready to execute. The processes involved in a potential interaction can actually *establish an interaction* by executing their completely compatible hybrid commands simultaneously. After interaction, they return to their local computing phases. Note that potential interactions involving a common process cannot be established at the same time.

The *hybrid guard scheduling problem* then is to establish interactions from the set of processes currently in their communication phases.

3 An Algorithm for the Hybrid Guard Scheduling Problem

Like the algorithm in [9], we assume an underlying network of processes that is completely connected via reliable, FIFO communication channels, and processes communicate exclusively by message passing. Furthermore, messages are timestamped; processes use the timestamps to maintain their logical clocks [12].

Each process p_i in the system concurrently executes two components, a *coordinator* and a *slave*. We will refer to each component separately as p_i whenever it is clear from the context. A coordinator communicates exclusively with slaves (and vice versa), attempting to *capture* slaves as participants of a potential interaction, while a slave waits for a captor, if any, to establish an interaction.

3.1 Coordinators

When process p_i enters its communication phase, it *activates* coordinator p_i, which immediately captures slave p_i. Then, coordinator p_i selects a hybrid command c, attempting to establish a potential interaction to execute c. (Slave p_i is informed of command c by coordinator p_i.) This potential interaction is built incrementally. Initially, the set of captured slaves P of this potential interaction consists of only p_i, and the set R of slaves to be captured consists of the processes named in c. Since potential interactions are dynamically structured, $P \bigcup R$ may not contain all the participants of the potential interaction p_i is attempting to establish. That is, there may be some participants that p_i may not yet know because p_i does not communicate with them directly within this potential interaction. However, due to the bi-lateral naming scheme—two processes must name each other in order to communicate—those participants that p_i does not know at this point are known to p_i's direct partners, or the direct partners of p_i's direct partners, and so on.

By sending a request message to each slave p_j in R, coordinator p_i attempts to capture slave p_j to select a hybrid command *compatible* with the commands selected by the slaves so far in P; compatible in the sense that a slave $p_k \in P$ wishes to communicate with p_j iff p_j wishes to communicate with it. Note that according to the disjointedness condition, p_j has at most one such hybrid command. By p_j selecting this command, p_i may learn of some other participants (slaves) that need to be captured. Therefore, when there are no more slaves that need to be captured (*i.e.*, $R = \emptyset$), the hybrid commands selected by slaves in P must be completely compatible. p_i then has succeeded in capturing all the participants of a potential interaction. It informs these participants to establish an interaction. However, if p_i cannot capture a slave in R, it aborts the attempt and *releases* all captured slaves except slave p_i. If p_i has another yet-to-be-selected hybrid command, it selects that command and attempts to establish another potential interaction. Otherwise, it releases slave p_i and stops its attempt to establish any interaction.

In order for coordinators to establish mutually disjoint interactions, a slave can be captured by only one coordinator at a time. Obviously, the requirements of different coordinators may conflict. These conflicts are resolved using unique *entry-times* obtained by each process from the contents of its logical clock when it entered its current communication phase. The larger the entry-time a process possesses, the younger the process is.

A *conflict* arises when a coordinator attempts to capture a slave that is already captured by another coordinator. Conflicts are resolved in favor of older coordinators in order to prevent a coordinator from being locked out from capturing slaves. That is, we let the older of the two coordinators capture/retain the slave under contention. The younger coordinator has then to wait hoping that the slave under contention will eventually be released by the older coordinator. Accordingly, a coordinator p_i may wait for more than one captured slave to be released by other older coordinators because p_i loses to the older coordinators for retaining those captured slaves. However, any slave lost by a coordinator p_i in a conflict will eventually be re-captured by p_i or will *reject* capture by p_i due to participation in another interaction. This is guaranteed since:

1. Deadlock cannot occur as old coordinators are not allowed to wait for captured slaves to be released by younger ones.
2. As other coordinators get younger and younger each time they are activated, p_i eventually will be old enough to win any contention.

When a coordinator p_i is waiting for a captured slave p_j to be released by an older coordinator p_k, we say that p_i's capture request has been *deferred* by p_j. p_i and p_j both maintain this "waiting for" relation until either one of the following happens:

- p_i aborts its coordination and *withdraws* its capture request to p_j.
- p_j's captor successfully establishes an interaction involving p_j. So p_j *rejects* p_i's capture request.
- p_j's captor releases p_j as it cannot establish an interaction involving p_j. So p_j *grants* p_i's capture request and is captured by p_i.

Note that a coordinator wishing to capture a slave sends to the slave only one capture request, which may be granted, rejected, or deferred by the slave. A capture request that is granted may need to be (re-)deferred later on in order to resolve a conflict.

3.2 Slaves

A slave p_j waits for its captor, if any, to establish an interaction involving p_j. If and when the captor releases p_j without establishing any interaction, p_j chooses a deferred capture request by the oldest coordinator and grants that capture request (so the other coordinators whose capture requests are deferred by p_j are still waiting for p_j to be released by a coordinator older than themselves). If, however, no capture request is deferred by p_j at this point, p_j simply idles waiting to be captured.

To reduce the message complexity of our algorithm which is dominated by the number of captures, we let slaves be captured by only younger coordinators. Therefore, only the youngest coordinator of a potential interaction can establish this interaction. This design choice also permits young coordinators to assemble interactions for which they may not have been ready when older coordinators attempted to establish one and failed.

In order to resolve conflicts while maintaining correctness, slave p_j uses the following strategy to handle a capture request by a coordinator p_i:

If p_j is not ready to join with p_i to establish an interaction attempted by p_i (*i.e.*, p_j has no hybrid command compatible with the commands selected by slaves so far captured by p_i), or p_j is younger than p_i, then p_j rejects p_i's request. Else, consider two subcases: (1) If p_j is not captured by any coordinator, then p_j grants p_i's request and is captured by p_i. (2) p_j is captured by a coordinator p_k. If p_k is older than p_i, then p_j remains captured by p_k while p_i's capture request is deferred. Otherwise, p_j asks p_k to release p_j. After p_k releases p_j, p_k's capture request is deferred by p_j. p_j then grants p_i's capture request.

According to this strategy and the initial condition that coordinator p_j captures slave p_j, slave p_j remains captured by coordinator p_j until coordinator p_j stops its attempt to establish any interaction. This is because any coordinator p_i that can capture slave p_j must be younger than it. If p_i is younger than slave p_j, it is also younger than coordinator p_j, and thus would not be able to capture slave p_j that is already captured by coordinator p_j.

The complete algorithms for coordinator p_i and slave p_i are given in Figures 2 and 3, respectively. They are presented as a CSP-like repetitive construct under the assumption of weak-fairness, *i.e.*, every command whose guard is continuously enabled is eventually executed. The messages communicated between coordinators and slaves are of the following types (assuming that p_i is the message sender and p_j is the recipient):

Request : p_i requests to capture p_j. The message also carries the entry-time of p_i and the set of commands selected by slaves so far captured by p_i.

Yes : p_i grants p_j's capture request. The message also carries p_i's selecting hybrid command.

No : p_i rejects p_j's capture request.

Release : p_i releases p_j.

Withdrawal : p_i withdraws its capture request to p_j.

Success : p_i informs p_j that an interaction involving p_j has been established.

Switch : p_i requests p_j to release p_i because an older coordinator attempts to capture p_i.

Switch_ok : p_i grants p_j's release request.

Due to the asynchrony of the system, messages from p_i to p_j may be *out-of-date* and thus must be discarded by p_j to maintain correctness. This happens to messages of *Yes*, *No*, and *Withdrawal* when p_i wants to grant/reject a deferred capture request by p_j, while p_j withdraws its capture request to p_i because some other slave has rejected capture by p_j. So, the message *Yes*/*No* becomes out-of-date when it arrives at p_j, and the message *Withdrawal* becomes out-of-date if p_i has already rejected p_j's capture request. Moreover, message *Switch* may be out-of-date when it arrives at p_j from p_i if p_j has established an interaction and sent out a *Success* message to p_i. Since messages between two processes are delivered in the order sent, a *Withdrawal* message from p_i to p_j is out-of-date iff p_j does not defer any capture request by p_i, and a *Switch* message is out-of-date iff p_j does not capture p_i. To detect whether a *Yes*/*No* message is out-of-date, a coordinator p_j attaches a unique id to each capture request it sends to p_i. (This id can simply be the timestamp of the request message.) p_i attaches this id to the *Yes*/*No* message in response to the capture request by p_j. p_j then can easily determine whether an arriving *Yes*/*No* message from p_i is out-of-date by comparing the received id with the id of the latest capture request to p_i.

4 Analysis of the Algorithm

We prove that the above algorithm satisfies the requisite safety and liveness properties.

Theorem 1 (Safety) *At any time the established interactions are mutually disjoint.*

Proof: A coordinator establishes an interaction by capturing slaves that correspond to all the participants of the interaction. Since a slave can be captured by one coordinator at a time, and a slave cannot be captured until it is released, no two coordinators can ever establish interactions with a common slave at the same time. □

```
1.1 $active$; $tried(select)$ $\longrightarrow$
        for $p_j, p_j \neq p_i, (id, p_j, c) \in P_i$ do
            send $Release$ to $p_j$;
        for $(id, p_j) \in D_i$ do
            send $Withdrawal$ to $p_j$;
        $D_i := \emptyset$;
        [ $\forall c \in H$ $tried(c)$ $\longrightarrow$
            send $Release$ to $p_i$;
            $P_i := \emptyset$;
            $active := 0$;
        $\square$ $\exists c \in H, \neg tried(c)$ $\longrightarrow$
            $select := c$;
            $P_i := \{(0, p_i, c)\}$;
            send $select$ to slave $p_i$ ] ;
1.2 $active$; $\neg tried(select)$; $D_i = \emptyset$;
            $recruit(P_i) \neq \emptyset$ $\longrightarrow$
        let $p_j \in recruit(P_i)$;
        let $id :=$ the current logical time;
        send $Request(id, t_i, P_i)$ to $p_j$;
        $D_i := D_i + (id, p_j)$;
1.3 $active$; $recruit(P_i) = \emptyset$ $\longrightarrow$
        for $(id, p_j, c_j) \in P_i$ do
            send $Success$ to $p_j$;
        $active := 0$;
```

```
1.4 receive $Yes(id, c)$ from $p_j$ $\longrightarrow$
        [ $(id, p_j) \in D_i$ $\longrightarrow$
            $P_i := P_i + (id, p_j, c)$;
            $D_i := D_i - (id, p_j)$;
        $\square$ /* out-of-date */
          else $\longrightarrow$ skip];
1.5 receive $No(id)$ from $p_j$ $\longrightarrow$
        [ $(id, p_j) \in D_i$ $\longrightarrow$
            $tried(select) := 1$;
            $D_i := D_i - (id, p_j)$;
        $\square$ /* out-of-date */
          else $\longrightarrow$ skip];
1.6 receive $Switch$ from $p_j$ $\longrightarrow$
        [ $\exists (id, p_j, c) \in P_i$; $recruit(P_i) \neq \emptyset$ $\longrightarrow$
            send $Switch\_ok$ to $p_j$;
            $P_i := P_i - (id, p_j, c)$;
            $D_i := D_i + (id, p_j)$
        $\square$ /* out-of-date */
          else $\longrightarrow$ skip];
```

t_i: p_i's entry-time.

H: the set of hybrid commands for which p_i is ready to execute. In particular, each command c is tagged with a Boolean value $tried(c)$, initialized to 0. $tried(c) = 1$ indicates that p_i has attempted to establish an interaction to execute c but failed.

active: a Boolean value indicating whether p_i is attempting to establish an interaction. It is initialized to 0 and is set to 1 when process p_i enters its communication phase.

select: the command p_i selects to attempt to establish an interaction. It is initialized to *nil* with $tried(nil) = 1$.

D_i: the set of elements of the form (id, p_j), representing that slave p_j has deferred p_i's capture request which has identity id. It is initialized to $\emptyset$.

P_i: the set of elements of the form (id, p_j, c_j), representing that p_j is captured to select its hybrid command c_j; id is the identity of p_i's latest capture request to p_j. The set is initialized to $\emptyset$.

$recruit(P_i)$: returns a set of processes that are not yet captured slaves of P_i, but are named in some command selected by a capture process of P_i.

Figure 2: Algorithm for coordinator p_i

```
2.1 receive select from $p_i$ ⟶ skip
2.2 receive Success from $p_j$ ⟶
      for $(id_k, p_k, t_k, c) \in queue$ do
          send $No(id_k)$ to $p_k$;
      $queue := \emptyset$;
      $H := \emptyset$;
      execute select;
2.3 receive Release from $p_j$ ⟶
      $captured := 0$;
2.4 receive Withdrawal from $p_j$ ⟶
      [ $p_j = captor \longrightarrow captured := 0$;
      □ $\exists\ (id, p_j, t_j, c) \in queue \longrightarrow$
          $queue := queue - (id, p_j, t_j, c)$;
      □ /* out-of-date */
        else ⟶ skip;
2.5 receive Switch_ok from $p_j$ ⟶
      $queue := queue + (captor, select)$;
      $captured := 0$;
2.6 receive $Request(id, t_j, P_j)$ from $p_j$ ⟶
      [ $t_i < t_j; \exists\, c_i \in H, compatible(c_i, P_j) \longrightarrow$
          $queue := queue + (id, p_j, t_j, c_i)$;
          let $captor = (id_k, p_k, t_k)$;
          [ $captured; t_j < t_k; \neg sw\_flag \longrightarrow$
              send Switch to captor;
              $sw\_flag := 1$
          □ else ⟶ skip ];
      □ else ⟶ send $No(id)$ to $p_j$ ];
2.7 $\neg\ captured; queue \neq \emptyset \longrightarrow$
      delete $(id_j, p_j, t_j, c)$ with the largest $t_j$
          from queue;
      send $Yes(id_j, c)$ to $p_j$;
      $select := c$;
      $captor := (id_j, p_j, t_j)$;
      $sw\_flag := 0$;
```

select: the command selected by p_i when it is captured.

captured: a Boolean value indicating whether slave p_i is captured or not. It is initialized to 0, and is set to 1 when process p_i enters its communication phase.

captor: a triple (id_j, p_j, t_j) representing that p_i is captured by p_j which has entry-time t_j, and the identity of the latest capture request by p_j is id_j. It is set to $(0, p_i, t_i)$ when process p_i enters its communication phase where it obtains entry-time t_i.

sw_flag: a Boolean value that is true iff p_i has sent a *Switch* to its current captor (because some old coordinator wishes to capture p_i). It is initialized to false.

queue: the set of elements of the forms (id, p_j, t_j, c), representing that the capture request (with identity id) by p_j, which has entry-time t_j, has been deferred by p_i; c is the command that p_i will select when it is captured by p_j. It is initialized to $\emptyset$.

compatible(c_i, P_j): returns a Boolean value indicating whether the hybrid command c_i is compatible with the commends in P_j, *i.e.*, $\forall (id, p_k, c_k) \in P_j,\ p_i \in c_k \iff p_k \in c_i$.

Figure 3: Algorithm for slave p_i

Lemma 1 *Whenever a coordinator p_i is activated, eventually, either it establishes an interaction involving slave p_i, or it sets* tried$(c), \forall c \in H$, *to false.*

Lemma 2 *Assume there exists a potential interaction among processes $p_1 \ldots p_m$. Let $c_1, \ldots, c_m$ be the completely compatible hybrid commands of processes $p_1, \ldots, p_m$, respectively. Then, the youngest coordinator of P, say p_m, can set* tried(c_m) *to false only if some other participant in P establishes an interaction.*

Theorem 2 (Liveness) *Eventually some process involved in a potential interaction will return to its local computing phase by establishing an interaction.*

Proof: Directly from Lemmas 1 and 2. □

We obtain an upper bound on the value of $\#(p_i[c])$, the number of messages generated by coordinator p_i in an attempt to establish an interaction to execute command c. These messages are either transmitted by p_i itself (*Request*, *Success*, *Release*, *Withdrawal*), or generated by other processes in response to p_i's messages (*Yes*, *No*, *Switch*, *Switch_ok*).

Theorem 3 $\#(p_i[c]) \leq 6(m-1)$, *where m is the number of participants in the potential interaction that p_i is attempting to establish.*

Proof: In an attempt to establish an interaction p_i can send at most $m-1$ *Requests*. A slave p_j replies *Yes*/*No* to a capture request by p_i. In the worst case, p_j is captured by some younger coordinator p_k while p_i is attempting to capture it. Then, p_i's capture request will cause p_j to send p_k a *Switch* message, p_k to answer *Switch_ok* to the message, and p_j to grant/reject p_k's deferred capture request. Also, p_i may send *Success*/*Release* to p_j if it captures p_j, or it may send *Withdrawal* to p_j if p_i's capture request has been deferred by p_j. □

We analyze the total number of messages that are generated by the coordinators as participants of an interaction. Assume there exists a potential interaction among $P = \{p_i \mid 1 \leq i \leq m\}$. If none of these processes in P establish interaction with other processes, then by Theorem 2, some coordinator p_i will establish this potential interaction. Let c_i be the hybrid command that p_i can execute in this potential interaction. In particular, the value

$$\#(P) = \sum_{1 \leq i \leq m} \#(p_i[c_i])$$

is a measure of how many messages coordinators of P will generate in order to establish this potential interaction. In our algorithm a coordinator can only capture slaves older than itself; that is, its capture request to a younger slave will always be rejected. Therefore, we have

$$\begin{aligned} \#(P) &\leq \Big(\sum_{1 \leq i < m} (6(i-1)+2) \Big) + 6(m-1) \\ &= (m-1)(3m+2) \end{aligned} \tag{1}$$

This overall message complexity would be nearly doubled if we would allow a coordinator to capture old slaves as well as young ones (still, conflicts are resolved in the same manner as in our algorithm).

5 Concluding Remarks

We have presented an efficient distributed algorithm for dynamically structured multiprocess communication in the context of Hybrid CSP. The algorithm improves upon an earlier implementation [9] in two significant ways: hybrid commands now may appear as guards, under a fairly general condition known as *disjointedness*; and the algorithm is an order of magnitude more efficient, message-wise, than the one in [9].

A number of algorithms, e.g. [17, 5, 2, 15, 11], have been proposed for statically structured multiprocess interaction. However, these algorithms cannot be extended in any straightforward way to the dynamic setting, since the underlying models assume that each participating process knows in advance the identities of all other participants.

References

[1] Back, R.J.R. and Kurki-Suonio, R. "Distributed cooperation with action systems," *ACM Trans. on Prog. Lang. and Systems*, Vol. 10, No. 4, 1988, pp. 513-544.

[2] Bagrodia, R. "Process synchronization: design and performance evaluation of distributed algorithms," *IEEE Trans. on Soft. Engr.*, Vol. 15, No. 9, Sep. 1989, pp. 1053-1065.

[3] Bolognesi, T. , Brinksma E. "Introduction to the ISO specification language LOTOS," *Computer Networks and ISDN Systems* , 14 (1987), pp. 25-59.

[4] Brinksma E. "On the design of Extended LOTOS — a specification language for open distributed systems," Ph.D. Dissertation, University of Twente, the Netherlands, 1988.

[5] Chandy, K.M and Misra, J. "Synchronizing asynchronous processes: the committee coordination problem," Technical Report, Dept. of Computer Science, Univ. of Texas, Austin, 1987.

[6] Forman, I. R. "On the design of large distributed systems," in *Proc. First Int'l Conf. on Computer Languages*, Miami, Florida, 1986, pp. 84-95.

[7] Francez, N. and Forman, I. R. *Interacting Processes: A Multiparty Approach to Coordinated Distributed Programming*. Forthcoming book, 1990.

[8] Hoare, C. A. R. "Communicating sequential processes," *Communications of the ACM* 21, 8 (Aug. 1978), pp.666-677.

[9] Joung, Y.H. and Smolka, S. A. "A completely distributed and message-efficient implementation of synchronous multiprocess communication," *Proc. of the 19th Int'l Conf. on Parallel Processing*, Aug. 1990.

[10] Joung, Y.H. "Tight complexity bounds on some multi-party synchronization problems," Technical Report No. 16, Department of Computer Science, SUNY at Stony Brook, 1990.

[11] Kumar, D. "An implementation of N-party synchronization using tokens," *Proc. of the 10th Int'l Conf. on Distributed Computing Systems*, 1990.

[12] Lamport, L. "Time, clocks and the ordering of events in a distributed system," *Communications of the ACM*, Vol.21, No.7, July 1978, pp. 558-565.

[13] Milne, G. J. "CIRCAL and the representation of communication, concurrency, and time," *ACM TOPLAS*, Vol. 7, No. 2, April 1985, pp. 270-289.

[14] Milner, R. "Calculi for synchrony and asynchrony," *Theoretical Computer Science* 25, 1983, pp. 267-310.

[15] Park, M. H. and Kim, M. "A distributed synchronization scheme for fair multi-process handshakes," *Information Processing Letters* 34, Apr. 1990, pp. 131-138.

[16] Ramesh, S. and Mehndiratta, S. L. "A new class of high-level programs for distributed computing systems," in *Proceedings of Fifth Conference on FST-TCS. Lecture Notes in Computer Science 206*, Springer-Verlag, Berlin, 1985, pp. 42-72.

[17] Ramesh, S. "A new and efficient implementation of multiprocess synchronization," in: *Proc. Conf. on PARLE, Lecture Notes in Computer Science 259*, Springer-Verlag, Berlin, 1987, pp. 387-401.

[18] Roman, G.-C. and Day M. S. "Multifaceted distributed systems specification using processes and event synchronization," *Proc. 7th Int'l Conf. on Software Engineering*, Orlando, Florida, March 26-29, 1984, pp. 44-55.

[19] U.S. Department of Defense, *Reference Manual for the ADA Programming Language*, New York, Springer-Verlag, 1983.

Cascaded GMW Sequences

A. Klapper, A. H. Chan, and M. Goresky
Northeastern University
Boston, MA, 02115

Abstract

Pseudorandom binary sequences with high linear complexity and low correlation function values are sought in many applications of modern communication systems. A new family of pseudorandom binary sequences, cascaded GMW sequences, is constructed. These sequences are shown to share many desirable correlation properties with the GMW sequences of Gordon, Mills, and Welch, for example high shifted autocorrelation values and, in many cases, three valued cross-correlation values with m-sequences. It is shown, moreover, that in many cases the linear complexities of cascaded GMW sequences are far greater than those of GMW sequences.

1 Introduction

Pseudorandom binary sequences with high linear complexity and low correlation function values are sought in many applications of modern communication systems. The class of binary "geometric sequences" based on odd primes [2] has recently attracted considerable attention due to their ease of generation using shift register hardware, and their enormous linear complexity. However it was shown recently [5] that the periodic autocorrelation function values of the geometric sequences (based on odd primes) is unacceptably high. On the other hand, the binary sequences of Gordon, Mills and Welch [4] have the same autocorrelation function as an m-sequence of the same period. Although the linear complexity of a GMW sequence is much greater than that of an m-sequence of the same period, it is desirable to achieve higher linear complexities.

In this paper we construct a new family of pseudorandom binary sequences which share many of the most desirable properties of both the geometric sequences and the GMW sequences. They have the same (periodic) autocorrelation function values as m-sequences of the same period, but in many cases have greater linear complexity than GMW sequences. They are balanced and can be generated using standard shift register hardware.

We will assume a basic understanding of finite fields and the trace function, since this material is very well explained in the excellent survey papers and books on the subject [3,7,6,8]. Let q be a fixed power of 2 and let $GF(q)$ denote the Galois field with q elements. If n is divisible by i, we denote the *trace function* from $GF(q^n)$ to $GF(q^i)$ by $Tr_{q^i}^{q^n}$. Let α be a primitive element of $GF(q^n)$. Suppose that $f : GF(q) \rightarrow GF(2)$ is a (possibly nonlinear) "feedforward" function. This gives rise to the *geometric sequence* [2] based on f and α,

$$\mathbf{S}_i = f(Tr^{q^n}_q(\alpha^i))$$

This is a very general class of pseudorandom binary sequences which includes the m-sequences (by taking $f(x) = Tr^q_2(x)$) and the GMW sequences (by taking $f(x) = Tr^q_2(x^k)$, where k is relatively prime to $q-1$). In other words, GMW(q^n, k, α) is the sequence

$$\mathbf{S}_i = Tr^q_2(Tr^{q^n}_q(\alpha^i)^k). \tag{1}$$

It is possible to generalize this construction if we have a tower of finite fields,

$$GF(2) \subset GF(q_1) \subset GF(q_2) \subset \ldots \subset GF(q_\ell)$$

by considering a nonlinear feedforward function $h : GF(q_\ell) \to GF(2)$ which is a composition of GMW-type functions, $h_i = Tr^{q_i}_{q_{i-1}}(x^{k_i})$ each of which drops down one step in the tower. We call such a composition $h = h_0 h_1 \cdots h_\ell$ a *cascade* and the resulting sequence $\mathbf{T}_i = h(\beta^i)$ is called a *cascaded GMW sequence.*

In this paper we show that a cascaded GMW sequence of period $q_\ell - 1$ has the same autocorrelation function as that of an m-sequence of the same period, and that the linear complexity of the cascaded sequence can be made to grow exponentially with the number of steps in the cascade. We also calculate the cross-correlation of a cascaded GMW sequence with an m-sequence of the same period, under the assumption that each of the steps $h_i : GF(q_i) \to GF(q_{i-1})$ is a quadratic function (that is, k_i has q_{i-1}-adic weight equal to 2). The techniques in this paper allow us to mix quadratic and linear functions, however the linear steps may all be removed without changing the sequence (as will be explained below). We find that the resulting cross-correlation function is three-valued and is low, just as in the case of Gold sequences and GMW sequences.

It is rather striking that such a computation can be made at all, because the cascaded feedforward function h is highly nonlinear and the complexity of the resulting sequence is enormous.

2 Definitions

In this section we give the formal definition of the cascaded GMW sequences and establish notation which will be used in the rest of the paper.

Let $n_1, \ldots, n_\ell$ be positive integers, $q_0 = 2$, and $q_i = q_{i-1}^{n_i}$ (for $i = 1, \ldots, \ell,$). Let $k_1, \ldots, k_{\ell-1}$ be positive integers satisfying $k_i < q_i$ and $\gcd(k_i, q_i - 1) = 1$. Let α be a primitive element of $GF(q_\ell)$.

Definition 1 *The sequence* $\mathbf{T}$ *whose* j*th term is*

$$\mathbf{T}_j = Tr^{q_1}_{q_0}(Tr^{q_2}_{q_1}(\cdots Tr^{q_\ell}_{q_{\ell-1}}(\alpha^j)^{k_{\ell-1}} \cdots)^{k_1}) \tag{2}$$

is called a cascaded GMW sequence.

If $\ell = 1$, the sequence $\mathbf{T}$ is an m-sequence of period $q_1 - 1$. If $\ell = 2$, the sequence $\mathbf{T}$ is a GMW sequence. These are special cases of a more general construction: Suppose $f : GF(q) \to GF(2)$ is a (possibly nonlinear) "feedforward" function.

Definition 2 *The geometric sequence based on* f *and* α *is the sequence*

$$\mathbf{S}_i = f(Tr^{q^n}_q(\alpha^i)) \tag{3}$$

3 Linear Complexity

In this section we study the linear complexity of a cascaded GMW sequence. In the case where each stage is quadratic, the linear complexity is completely determined. We will make use of the following result of Brynielsson [1] which applies to any geometric sequence:

Proposition 1 *Suppose the geometric sequence* $\mathbf{S}$ *of equation (3) is generated by the feedforward function* $f : GF(q) \to GF(2)$ *which has a polynomial representation*

$$f = \sum_{i=0}^{q-2} A_i x^i$$

with coefficients A_i *in* $GF(q)$*. Then the linear complexity of the sequence* $\mathbf{S}$ *is equal to* $\sum_{A_i \neq 0} n^{||i||}$*, where* $||i||$ *denotes the dyadic weight of the integer* i *(i.e. the number of ones in the base 2 representation of* i*).*

As a consequence of this result, we must determine which monomials appear with nonzero coefficients in $g_{\ell-1}(x) = Tr_{q_0}^{q_1}(Tr_{q_1}^{q_2}(\cdots Tr_{q_{\ell-2}}^{q_{\ell-1}}(x^{k_{\ell-1}})^{k_{\ell-2}} \cdots)^{k_1})$.

Theorem 1 *Suppose that, for each* i, $k_i = q_{i-1}^{s_i} + q_{i-1}^{t_i}$ *(i.e.,* k_i *has* q_{i-1}*-adic weight two), with* $0 \leq s_i < t_i < n_i$ *and* $n_i / \gcd(n_i, t_i - s_i)$ *odd. Then the linear complexity of the cascaded GMW sequence* $\mathbf{T}$ *is*

$$n_1 n_2^2 n_3^4 \cdots n_\ell^{2^{\ell-1}}.$$

Proof Sketch: The polynomial representation of $g_{\ell-1}$ is derived by alternately raising to a k_ith power and applying a trace function. We show that, when this is done, there is no cancellation of terms each time we raise to a power or apply a trace function. Raising to a k_ith power thus squares the number of monomials, while applying a trace function multiplies the number of monomials by the degree of the field extension. This implies that the number of monomials in the final feedforward function is

$$n_1 n_2^2 n_3^4 \cdots n_{\ell-1}^{2^{\ell-2}}.$$

Moreover, we show that raising to a k_ith power doubles the dyadic weight of the degree of each monomial, while applying a trace function leaves these dyadic weights unchanged. Thus the dyadic weight of the degree of each monomial in the final feedforward function is $2^{\ell-1}$. The theorem follows from the preceding proposition. □

Now suppose we wish to restrict the complexity of the feedforward functions by requiring all exponents to have dyadic weight two. Consider a tower of fields with $n_1 = 3$, and $n_2 = \cdots = n_\ell = 2$. The highest linear complexity we can achieve for a GMW sequence with these restrictions is $3 \cdot 2^\ell$, which occurs for the sequence with feedforward function

$$Tr_2^{2^{3 \cdot 2^{\ell-2}}}(x^3).$$

In contrast, if we take a cascaded GMW sequence over this tower of fields, with $k_i = 1 + q_{i-1}$, then by Theorem 1, the linear complexity is

$$3 \cdot 2^2 \cdot 2^4 \cdots 2^{2^{\ell-1}} = 3 \cdot 2^{2^\ell - 2},$$

which is exponentially larger. (Note that we are forced to take $n_1 = 3$ in order to obtain a quadratic exponent, since over $GF(4)$ we have $Tr_2^4(x^3) = 0$.)

As a second example, suppose we leave the exponents unrestricted. Consider a tower of fields with $n_1 = n_2 = \cdots = n_\ell = 3$, so $q_i = 2^{3^i}$. Thus the periods of the sequences generated is at most $2^{3^\ell} - 1$. The largest possible linear complexity for a GMW sequence with this period is $3^{3^{\ell-1}+\ell-2}$. This occurs for the sequence with feedforward function

$$Tr_2^{2^{3^{\ell-1}}}(x^{2^{(3^{\ell-1}-1)}-1}).$$

Now consider a cascaded GMW sequence with $k_i = 1 + q_{i-1}$ for $i = 1, \cdots, \ell - 2$, and $k_{\ell-1}$ the solution to the congruence

$$k_{\ell-1} \prod_{i=0}^{\ell-1} (1 + q_i) \equiv 2^{(3^{\ell-1}-1)} - 1 \bmod 2^{3^{\ell-1}} - 1.$$

It follows that the product of the $k_i's$ has $q_{\ell-1}$-adic weight $3^{\ell-1} - 1$, which is the maximum weight exponent that can contribute to the final feedforward function. Note that $\gcd(2^{3^i} + 1, 2^{3^{\ell-1}} - 1) = 1$, so this congruence can be solved (in fact uniquely).

Let

$$g(x) = Tr_2^{2^3}(Tr(_{2^3}^{2^9}(\cdots Tr_{2^{3^{\ell-2}}}^{2^{3^{\ell-1}}}(x)^{k_{\ell-2}} \cdots)^{k_2})^{k_1}).$$

The feedforward function defined by these k_i is $f(x) = g(x^{k_{\ell-1}})$. The argument in the proof of Theorem 1 shows that there is no cancellation of terms in $g(x)$ as we perform the compositions of traces and exponentiations. Moreover, $k_{\ell-1}$ is relatively prime to $q_{\ell-1} - 1$, so raising to the $k_{\ell-1}$ power is a permutation of $GF(q_{\ell-1})$. It follows that composing $g(x)$ with $x^{k_{\ell-1}}$ simply changes the exponents, that is, does not introduce any cancellation. Thus all terms in $f(x)$ remain distinct.

We also know that f maps $GF(q_{\ell-1})$ to $GF(2)$. It follows from Galois theory that if f has a term x^u, then it also has every term x^v where $v \equiv 2^j u (\bmod q_{\ell-1} - 1)$ for some j. By construction, f has the term x^u, $u = 2^{(3^{\ell-1}-1)} - 1$, and hence has all $3^{\ell-1}$ terms whose exponents have dyadic weight $3^{\ell-1} - 1$. These terms contribute $3^{3^{\ell-1}+\ell-2}$ to the linear complexity, and this is precisely the linear complexity of the GMW sequence we discussed above. Thus the linear complexity of this cascaded sequence exceeds the linear complexity of the GMW sequence by $\sum 3^{||j||}$, where the sum ranges over the remaining $3^{2^{\ell-1}-1} - 3^{\ell-1}$ terms, hence by at least $3^{2^{\ell-1}-1} - 3^{\ell-1}$.

4 Autocorrelation Functions

Recall that the (periodic) cross-correlation function of two binary sequences **S** and **T** of period $q^n - 1$ is defined as

$$\Theta_{\mathbf{S},\mathbf{T}}(\tau) = \sum_{r=1}^{q^n-1} (-1)^{\mathbf{S}_{r+\tau}} (-1)^{\mathbf{T}_r}.$$

and the autocorrelation function of the sequence **S** is defined as $A_{\mathbf{S}}(\tau) = \Theta_{\mathbf{S},\mathbf{S}}(\tau)$. For any geometric sequence, $\mathbf{S}_i = f(Tr_q^{q^n}(\beta^i))$ the authors previously showed [5] that the autocorrelation function is given by

$$A_{\mathbf{S}}(\tau) = q^{n-1} c_f(t) - 1 \quad \text{if } \tau = tv, \tag{4}$$

$$A_{\mathbf{S}}(\tau) = q^{n-2} I(f)^2 - 1 \quad \text{otherwise.} \tag{5}$$

where the imbalance $I(f) = \sum_{u \in GF(q)} (-1)^{f(u)}$, $v = (q^n - 1)/(q-1)$ and $c_f(t)$ is the autocorrelation function of f,

$$c_f(t) = \sum_{u \in GF(q)} (-1)^{f(u)} (-1)^{f(\beta^t u)}.$$

Here, $\beta = \alpha^v$ is the corresponding primitive element of $GF(q)$.

Generalizing a well known result on the autocorrelation of a GMW sequences, we now apply this result inductively to cascaded GMW sequences, to obtain the following.

Proposition 2 *The autocorrelation function of the cascaded GMW sequence 1 is given by*

$$A_{\mathbf{S}}(\tau) = \begin{cases} q_\ell - 1 & \text{if } \tau = 0 \\ -1 & \text{otherwise.} \end{cases}$$

5 Cross-Correlation Functions

In this section we give estimates and exact values for the periodic cross-correlation functions of certain cascaded GMW sequences. The authors previously calculated the periodic cross-correlation function for "quadratically related" geometric sequences [5]. These results are summarized here. **S** and **T** denote geometric sequences with feedforward functions $f, g : GF(q) \to GF(2)$ respectively, based on primitive elements $\alpha, \beta \in GF(q^n)$ respectively. We let $F(u) = (-1)^{f(u)}$, $G(u) = (-1)^{g(u)}$, and $\Delta_a(f,g) = \sum_{u \in GF(q)} F(au)G(u^2)$ for any nonzero a in $GF(q)$.

Theorem 2 *[5] Suppose $\beta = \alpha^k$, with $k = q^i + q^j$ (i.e.,* **S** *and* **T** *are quadratically related). Let $m = n - \gcd(n, j-i)$. Then m is even and the values for the cross-correlation function $\Theta_{\mathbf{S},\mathbf{T}}(\tau)$ are:*

1. $\Theta_{\mathbf{S},\mathbf{T}}(\tau) = q^{n-2} I(f) I(g) - F(0)G(0)$ *which occurs for* $q^n - q^{m+1} + q^m - 1$ *values of* τ.
2. $\Theta_{\mathbf{S},\mathbf{T}}(\tau) = (q^{n-2} - q^{n-m/2-2}) I(f) I(g) + q^{n-m/2-1} \Delta_a(f,g) - F(0)G(0)$ *which occurs for* $(q^m + q^{m/2})/2$ *values of* τ *for each nonzero* $a \in GF(q)$.
3. $\Theta_{\mathbf{S},\mathbf{T}}(\tau) = (q^{n-2} + q^{n-m/2-2}) I(f) I(g) - q^{n-m/2-1} \Delta_a(f,g) - F(0)G(0)$ *which occurs for* $(q^m - q^{m/2})/2$ *values of* τ *for each nonzero* $a \in GF(q)$.

We let ℓ, $\{n_i\}$, $\{q_i\}$, $\{k_i\}$ determine a cascaded GMW sequence **T** with primitive element β, as in equation 2. Observe that such a cascaded GMW sequence is a geometric sequence with feedforward function

$$f(x) = Tr_{q_0}^{q_1}(Tr_{q_1}^{q_2}(\cdots Tr_{q_{\ell-1}}^{q_\ell}(x)^{k_{\ell-1}} \cdots)^{k_1}).$$

Thus Theorem 2 can be applied to cascaded GMW sequences whose underlying m-sequences are quadratically related. The feedforward function f is balanced (i.e., $I(f) = 0$), so we obtain the following estimates for the cross-correlation of two cascaded GMW sequences.

Theorem 3 *Let* **T** *and* **S** *be cascaded GMW sequences over a tower of fields (with parameters as in the definition) and based on primitive elements* α *and* β *respectively. Suppose* $\beta = \alpha^k$, *where* $k = q_{\ell-1}^i + q_{\ell-1}^j$ *(and the exponents* k_i, $i \leq \ell - 1$ *not necessarily of* q_{i-1}*-adic weight two). Then*

$$|\Theta_{\mathbf{S},\mathbf{T}}(\tau) + 1| \leq q_{\ell-1}^{\frac{1}{2}(n_\ell + \gcd(n_\ell, j-i))}.$$

In the special case of an m-sequence and a cascaded GMW sequence for which each k_i has q_{i-1}-adic weight two, we can apply Theorem 2 inductively to compute the cross-correlation exactly. The critical observation is that the quantity $\Delta_a(f, g)$ differs by one from the cross-correlation function of an m-sequence and a cascaded GMW sequence with one fewer stage. As with GMW sequences, the cross-correlation function is three valued.

Theorem 4 *Let* **T** *be a cascaded GMW sequence, and let* **S** *be the m-sequence* $Tr_2^{q_\ell}(\alpha^{i-1})$, *where* α *is a primitive element of* $GF(q_\ell)$ *over* $GF(2)$. *Let* $\beta = \alpha^{k_\ell}$. *Assume that, for each* $i \geq 1$, $k_i = q_{i-1}^{s_i} + q_{i-i}^{t_i}$ *for* $0 \leq s_i \leq t_i < n_i$, $s_1 < t_1$, *i.e.,* k_i *has* q_{i-1}*-adic weight 2. Assume moreover that, for each* i, $n_i / \gcd(n_i, t_i - s_i)$ *is odd (i.e.,* $\gcd(k_i, q_i - 1) = 1$*). Let* $m_i = n_i - \gcd(n_i, t_i - s_i)$, $d_i = n_1 n_2 \cdots n_i$ $(d_0 = 1)$, *and* $e_i = \sum_{j=1}^{i} d_{j-1} m_j$. *Then the cross-correlation function of* **S** *and* **T** *is given by*

$$\Theta_{\mathbf{S},\mathbf{T}}(\tau) = \begin{cases} -1 & \textit{occurring} \quad 2^{d_\ell} - 2^{e_\ell} - 1 \quad \textit{times} \\ 2^{d_\ell - e_\ell/2} - 1 & \textit{occurring} \quad (2^{e_\ell} + 2^{e_\ell/2})/2 \quad \textit{times} \\ -2^{d_\ell - e_\ell/2} - 1 & \textit{occurring} \quad (2^{e_\ell} - 2^{e_\ell/2})/2 \quad \textit{times.} \end{cases}$$

References

[1] L. Brynielsson "On the linear complexity of combined shift registers," Proceedings of Eurocrypt 1984, 156-160.

[2] A. H. Chan and R. Games, "On the linear span of binary sequences from finite geometries, q odd," Proceedings of Crypto 1986, 405-417.

[3] S. Golomb, *Shift Register Sequences.* Laguna Hills, CA: Aegean Park Press, 1982.

[4] B. Gordon, W. H. Mills, and L. R. Welch, "Some new difference sets," *Canad. J. Math.* **14** (1962) 614-625.

[5] A. Klapper, and A.H. Chan, M. Goresky, *Cross-Correlations of Geometric Sequences in Characteristic 2 and GMW Sequences*, Northeastern Univ. Tech. Report NU-CCS-90-12 (1990).

[6] R. Lidl and H. Niederreiter *Finite Fields* in *Encyclopedia of Mathematics vol 20* Cambridge University Press, Cambridge (1983).

[7] R. McEliece *Finite Fields for Computer Scientists and Engineers.* Kluwer Academic Publishers, Boston 1987.

[8] M. Simon, J. Omura, R. Scholtz, and B. Levitt, *Spread-Spectrum Communications.* Volume 1, Computer Science Press, 1985.

AN N-USER MULTIPLE ACCESS SYSTEM BASED ON $(N, 1, m)$ CONVOLUTIONAL CODES

Vivek Telang and Mark A. Herro

Tellabs Research Center, Mishawaka, IN 46545.

Department of Electrical and Computer Engineering,
University of Notre Dame, Notre Dame, IN 46556.

ABSTRACT

This paper describes a new approach to designing an N-user Multiple-Access System for a binary channel. We design a coding scheme for N sources, based on convolutional codes, such that when all N sources are sending data, the channel is used at its maximum capacity. However, when fewer than N sources are active, the unused channel capacity is used to achieve error control. The extent of error control is inversely proportional to T, the number of active sources. Possible applications of these codes are in a multiple-access system used by data sources as well as voice sources, since the codes can exploit the inactivity of bursty voice sources to afford error-protection for the data sources without requiring additional channel bandwidth. We present design principles and also describe constructive techniques to design uniquely-decodable and non-catastrophic coding schemes for such multi-user systems. We describe in detail a 2-user system, and outline techniques to design systems with a greater number of users. In particular, we present a constructive algorithm for the design of codes for $N-$user systems, where $N = 2^r$, where r is an integer. We also determine the error-control capabilities of these systems in terms of T, the number of active sources.

I. INTRODUCTION

Efficient use of transmission facilities often requires the sharing of resources by a number of users. Commonly used methods of sharing resources are Time Division Multiplexing (TDM) and Frequency Division Multiplexing (FDM). In this paper, we describe a novel approach to designing an N-user Multiple-Access System for a Binary Channel. We design a coding scheme for N sources, such that when all N sources are *active*, *i.e.*, sending data, the channel is used at its maximum capacity. In this mode, the bandwidth efficiency of the multiple access system is the same as that of standard TDM. However, when only T sources are active ($T < N$), the scheme has an advantage over TDM. In the standard TDM transmission system, each source is assigned a fixed time-slot for its data. When a source does not have any data to transmit, its slot is unused, resulting in a waste of channel capacity. In the proposed multiple-access system, the bandwidth that is not being used by the $N - T$ inactive sources is used by the T active sources to achieve error control. The extent of error control is inversely proportional to T, the number of active sources. The codes used by each of the N sources are independent of T, which implies that each source does not need to know the status (activity/inactivity) of the other $N - 1$ sources.

This paper is organized in the following manner. In the next section we briefly describe the problem of coding for multi-user systems, and outline the proposed solution. In section III, we present a detailed example which describes how the proposed technique can be applied to design of a 2-user system. In section IV we describe the problems involved in designing codes for systems where the number of users N, is greater than 2, and present a constructive method for the special case when N is an integral power of 2, *i.e.*, $N = 2^r$ where r is an integer. The final section presents results and directions for future research.

II. BACKGROUND AND SOLUTION OUTLINE

The fundamental problem in designing a multi-user system is to be able to assign to each user a set of signals such that when the composite formed by combining one signal from each set is transmitted over the channel, the receiver can recover the individual signals from the composite signal. In an $N-$user system, N sources produce information to be transmitted over a common communication channel, the message generated by the i^{th} source being denoted by m_i. The sources are assumed to be statistically independent and the channel is assumed to be memoryless. Each message is assumed to be a stream of information symbols occurring at some fixed rate, say R_i symbols/second for the i^{th} user. The N information streams are converted via a device, which we refer to as the "contention resolver" into L signals $x_1, x_2, \cdots, x_L$, which are transmitted over the channel. Depending on the particular multi-user system, the contention resolver may be implemented as a set of encoders, modulators, and/or some type of a multiplexor. In some situations, L is equal to 1, in which case all N data streams enter a common device which arranges for all N information streams to be carried by a single signal. An example of such a contention resolver is the multiplexor in a TDMA system. In other situations, L is equal to N, and each message is individually processed, resulting in $L = N$ different signals which are transmitted over the channel. The combining of these N signals then occurs on the channel itself. An example of such a system is the Mulitple Access Channel (MAC). There can be many situations in between, *i.e.*, $1 < L < N$, for example the "T Active users out of N Multiple Access Channel (TANMAC)" described by Mathys [1].

In past research, we have designed multi-user systems with dynamic error control based on cyclic block codes such as the BCH codes and the Reed-Solomon codes [2]. In this paper we propose an N-user multiple access system based on convolutional codes. Convolutional codes seem attractive for this purpose because of the extensive study that has gone into designing good convolutional codes. In the proposed scheme each of the N sources is assigned a rate $1/N$ convolutional code. We denote the code assigned to source S_i by C_i, $1 \leq i \leq N$, and the free distance of code C_i by d^i_{free}. We represent the information sequence from the i^{th} source S_i by the sequence

$$u^i = [u^i_0, u^i_1, u^i_2, \ldots], \tag{1}$$

where the subscript denotes the time index. This sequence is the input to the encoder of a $(N, 1, m)$ convolutional code C_i, and the encoder output is therefore the sequence of $N-$tuples

$$\boldsymbol{x}^i = [\boldsymbol{x}^i_o, \boldsymbol{x}^i_1, \boldsymbol{x}^i_2, \cdots] \tag{2}$$

where $\boldsymbol{x}^i_l$ is the $N-$tuple of output symbols from the i^{th} encoder at time l. We define the transfer function matrix for code C_i to be the $(1 \times N)$ matrix

$$g_i(D) = \left[\; g^1_i(D) \;\; g^2_i(D) \;\; \cdots \;\; g^N_i(D) \;\right]. \tag{3}$$

The $D-$transform of the output of the i^{th} encoder is thus given by

$$\boldsymbol{x}^i(D) = u^i(D)\boldsymbol{g}_i(D), \tag{4}$$

where $u^i(D)$ is the $D-$transform of the information sequence u^i, $1 \leq i \leq N$.

Since there are N sources, and since the information sequence from each source is processed by a $(n,k,m) = (N,1,m)$ convolutional code[1], we have N sequences of $N-$tuples, $\boldsymbol{x}^i$, $1 \leq i \leq N$. If we wish to transmit these sequences over a single-user binary channel, we need to somehow resolve the contention between the N sequences. We achieve this by performing a component-wise *modulo-2* sum of the N $N-$tuples at each time instant, to produce the output sequence $\boldsymbol{v} = [\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots]$, where

$$\begin{aligned} \boldsymbol{v}_l &= \boldsymbol{x}_l^1 \oplus \boldsymbol{x}_l^2 \oplus \cdots \oplus \boldsymbol{x}_l^N \\ &= \bigoplus\!\!\!\!\sum_{i=1}^{N} \boldsymbol{x}_l^i, \qquad l = 1,2,\ldots \end{aligned} \tag{5}$$

and $\bigoplus\!\!\!\!\sum$ denotes a *modulo-2* summation. In the $D-$transform domain,

$$\begin{aligned} \boldsymbol{v}(D) &= \boldsymbol{v}_0 + \boldsymbol{v}_1 D + \boldsymbol{v}_2 D^2 + \cdots, \\ &= \bigoplus\!\!\!\!\sum_{i=1}^{N} \boldsymbol{x}^i(D) && (6) \\ &= \bigoplus\!\!\!\!\sum_{i=1}^{N} u^i(D)\boldsymbol{g}_i(D). && (7) \end{aligned}$$

Rewriting (7) in matrix notation, we get

$$\boldsymbol{v}(D) = \boldsymbol{u}(D)\boldsymbol{G}(D), \tag{8}$$

where $\boldsymbol{u}(D)$ is the $N-$dimensional vector $\begin{bmatrix} u^1(D) & u^2(D) & \ldots & u^N(D) \end{bmatrix}$ and $\boldsymbol{G}_N(D)$ is the $(N \times N)$ matrix represented by

$$\boldsymbol{G}_N(D) = \begin{bmatrix} \boldsymbol{g}_1(D) \\ \boldsymbol{g}_2(D) \\ \vdots \\ \boldsymbol{g}_N(D) \end{bmatrix} = \begin{bmatrix} g_1^1(D) & g_1^2(D) & \ldots & g_1^N(D) \\ g_2^1(D) & g_2^2(D) & \ldots & g_2^N(D) \\ \vdots & \vdots & & \vdots \\ g_N^1(D) & g_N^2(D) & \ldots & g_N^N(D) \end{bmatrix}, \tag{9}$$

where the subscript in $\boldsymbol{G}_N(D)$ refers to the number of users. We observe that (8) is precisely the representation of a convolutional coding system with N inputs and N outputs, *i.e.*, a rate N/N convolutional code.

We now consider the situation where T of the N sources are active, and $T \leq N$. Clearly, when $T = N$, the N input information bits are encoded into N channel bits, and the information rate is unity, which is the capacity of the noiseless binary channel. However, if fewer than N sources are active, *i.e.*, $T < N$, the system can be viewed as a (N,T,m) convolutional coding system. For the sake of simplicity, and without any loss of generality, we assume that the active sources are the sources $S_1, S_2, \ldots, S_T$, and that the remaining sources, *i.e.*, sources $S_{T+1}, S_{T+2}, \ldots, S_N$ are inactive. We assume that the information bits of the $N-T$ inactive sources are zeroes, *i.e.*,

---

[1] Here an (n,k,m) convolutional code implies that k information sequences are encoded into n channel sequences using k shift registers, each of length less than or equal to m.

$$u^{T+1} = u^{T+2} = \cdots = u^N = 0. \tag{10}$$

The corresponding encoder outputs are given by

$$\boldsymbol{x}^{T+1} = \boldsymbol{x}^{T+2} = \cdots = \boldsymbol{x}^N = 0. \tag{11}$$

Combining (5) and (11), we see that the channel code vector at any time instant l is given by

$$\begin{aligned} v_l &= \boldsymbol{x}_l^1 \oplus \boldsymbol{x}_l^2 \oplus \cdots \oplus \boldsymbol{x}_l^T \\ &= \sum_{i=1}^{T} \boldsymbol{x}_l^i, \qquad l = 1, 2, \ldots \end{aligned} \tag{12}$$

In the $D-$transform domain,

$$v(D) = \sum_{i=1}^{T} u^i(D) g_i(D), \tag{13}$$

and consequently the transfer function matrix can be represented by the $(T \times N)$ matrix

$$G_T(D) = \begin{bmatrix} g_1(D) \\ g_2(D) \\ \vdots \\ g_T(D) \end{bmatrix} = \begin{bmatrix} g_1^1(D) & g_1^2(D) & \ldots & g_1^N(D) \\ g_2^1(D) & g_2^2(D) & \ldots & g_2^N(D) \\ \vdots & \vdots & & \vdots \\ g_T^1(D) & g_T^2(D) & \ldots & g_T^N(D) \end{bmatrix}, \tag{14}$$

which is the transfer function matrix of a (N, T, m) convolutional code. The free distance d_{free} of the composite multi-user code C generated by the transfer function matrix $G_T(D)$ depends on the individual codes C_i used. The coding system will, by design, have certain error-control capabilities. Thus we have succeeded in using the channel capacity left unused by the $N-T$ inactive sources to achieve error control for the T active sources.

The problem is now reduced to that of designing a (N, N, m) unity rate convolutional coding system, comprised of N rate $(1/N)$ convolutional codes with known error-control capabilities. The two requirements we impose on the design are that all channel bit sequences be uniquely decodable, and not susceptible to catastrophic error propagation regardless of the number of active sources. In other words we propose to design a (N, N, m) convolutional coding scheme with a composite transfer function matrix defined by (9) such that the codes generated by any $T-$row subset of the $(N \times N)$ composite matrix also generates a uniquely decodable and non-catastrophic code, for $1 \leq T \leq N$. In addition, our design objective is that when the number of active sources T is less than N, the resulting code should have some error-control capabilities. We use the results due to Massey and Sain [3] and Forney [4] to help us achieve this. Interesting observations about the *modulo-2* sum of the output of convolutional codes have been reported by Peterson and Costello in [5] and we use these results to choose the N codes that constitute the multiple-access system. We also use results reported by Massey *et al* in [6] and Bahl and Jelinek in [7] to choose individual codes C_i with large free distances.

III. EXAMPLE: A 2-USER SYSTEM

We illustrate our method by means of a simple example. Consider an $N = 2$-user system. We propose to assign each of the two sources a rate (1/2) convolutional code. Each source encoder outputs a 2-bit code vector for each information bit. The contention resolver then performs a *modulo-2* sum of the output of the encoders, and transmits the result over the channel. Thus, the channel bits are the output of a $(2, 2, m)$ convolutional code, where the two information bits come from the two sources. The number of transmitted bits over the channel is two channel bits per two information bits and the transmission rate is therefore 1 bit/channel use, which is the capacity of the noiseless binary channel. What is important to note here is that when one of the two sources is inactive, *i.e.*, not transmitting information, the system reduces to a rate (1/2) convolutional coding system as far as the remaining (one) active source is concerned.

We now address the problem of designing a $(2, 2, m)$ unity rate convolutional coding scheme with a generator transfer function matrix $G_2(D)$ given by

$$G_2(D) = \begin{bmatrix} g_1(D) \\ g_2(D) \end{bmatrix} = \begin{bmatrix} g_1^1(D) & g_1^2(D) \\ g_2^1(D) & g_2^2(D) \end{bmatrix}, \tag{15}$$

where the subscript in $G_2(D)$ refers to N, the total number of sources. Clearly, if only source S_1 is active, the code reduces to

$$G_2^1(D) = \begin{bmatrix} g_1(D) \end{bmatrix} = \begin{bmatrix} g_1^1(D) & g_1^2(D) \end{bmatrix}, \tag{16}$$

whereas if only source S_2 is active, the code reduces to

$$G_2^2(D) = \begin{bmatrix} g_2(D) \end{bmatrix} = \begin{bmatrix} g_2^1(D) & g_2^2(D) \end{bmatrix}. \tag{17}$$

We require that the codes generated by any of the three transfer function matrices $G_2(D)$, $G_2^1(D)$ or $G_2^2(D)$ be uniquely decodable and non-catastrophic. In addition, our design objective is that the rate 1/2 codes generated by either $G_2^1(D)$ or $G_2^2(D)$ should have some error-control capabilities. The remainder of this section describes a construction technique for the design of $G_2(D)$ that satisfies all the above requirements.

We start with a $(2, 1, m)$ convolutional code with known distance properties, say $G(D) = \begin{bmatrix} g^1(D) & g^2(D) \end{bmatrix}$, and assign it to source S_1. Thus,

$$G_2^1(D) = \begin{bmatrix} g^1(D) & g^2(D) \end{bmatrix}, \tag{18}$$

We now propose to assign to source S_2 a code that is a "cyclic shift" of the above code. Thus,

$$G_2^2(D) = \begin{bmatrix} g^2(D) & g^1(D) \end{bmatrix}, \quad \text{and} \tag{19}$$

$$G_2(D) = \begin{bmatrix} g^1(D) & g^2(D) \\ g^2(D) & g^1(D) \end{bmatrix}. \tag{20}$$

We now determine the conditions for this code to be uniquely decodable and non-catastrophic. Massey and Sain [3] have shown that a (n, k, m) convolutional code defined by a transfer function matrix $G(D)$ is *non-catastrophic* if and only if

$$\mathrm{GCD}[\Delta_i(D) : i = 1, 2, \ldots, \binom{n}{k}] = D^l \tag{21}$$

for some $l \geq 0$. Here, GCD denotes the greatest common divisor, and $\Delta_i(D)$, $i = 1, 2, \ldots, \binom{n}{k}$, are the determinants of the $\binom{n}{k}$ distinct $(k \times k)$ submatrices of the transfer function matrix $G(D)$. In our case, since $n = k = N$, this reduces to the requirement that

$$\Delta_1(D) = D^l, \text{ for some } l \geq 0, \tag{22}$$

where $\Delta_1(D)$ is the determinant of the matrix $G_2(D)$. For the case $N = K = 2$, dropping the subscript, we have

$$\begin{aligned} \Delta(D) &= [g^1(D)]^2 + [g^2(D)]^2 && (23) \\ &= [g^1(D^2)] + g^2(D^2)]. && (24) \end{aligned}$$

Now, if we choose $g^1(D)$ and $g^2(D)$ such that

$$g^2(D) = g^1(D) + D^p \tag{25}$$

where $p \geq 0$ is an arbitrary integer, then

$$\begin{aligned} \Delta(D) &= [g^1(D^2)] + g^2(D^2)] && (26) \\ &= D^{2p}. && (27) \end{aligned}$$

Thus this 2-user code is uniquely decodable and non-catastrophic. Since $G_2^2(D)$ is simply a cyclic shift of $G_2^1(D)$, the error control capabilities of $G_2^1(D)$ and $G_2^2(D)$ are identical. Thus, each of the sources is afforded the same degree of error control when the other source is inactive.

We illustrate this construction by means of a simple example. We pick $G(D)$ to be the transfer function matrix of the (2,1,2) code with the best d_{free} [8], and assign it to source S_1. Thus,

$$G_2^1(D) = \begin{bmatrix} 1 + D^2 & 1 + D + D^2 \end{bmatrix}. \tag{28}$$

This code has $d_{free} = 5$. We assign a cyclic shift of $G_2^1(D)$ to source S_2. Thus,

$$G_2^2(D) = \begin{bmatrix} 1 + D + D^2 & 1 + D^2 \end{bmatrix}. \tag{29}$$

The composite transfer function matrix is given by

$$G_2(D) = \begin{bmatrix} 1 + D^2 & 1 + D + D^2 \\ 1 + D + D^2 & 1 + D^2 \end{bmatrix}. \tag{30}$$

It is easy to see that the determinant of $G_2(D)$ is given by

$$\begin{aligned} \Delta(D) &= (1 + D^2) \times (1 + D^2) + (1 + D + D^2) \times (1 + D + D^2) && (31) \\ &= 1 + D^4 + 1 + D^2 + D^4 && (32) \\ &= D^2. && (33) \end{aligned}$$

The composite code is therefore uniquely decodable and non-catastrophic. When either of the 2 sources is inactive, the information from the active source is effectively encoded by a (2,1,2) convolutional code with $d_{free} = 5$. These results are tabulated in Table 1.

N	m	T	d_{free}
2	2	1	5
2	2	2	1
3	2	1	6
3	2	2	4
3	2	3	1
4	2	1	9
4	2	2	4
4	2	3	2
4	2	4	1

Figure 1: Relationship between T, the number of active sources, and d_{free}, the free distance of the code for $N = 2, 3$ and 4.

IV. EXTENSION TO AN N-USER SYSTEM

We now consider codes for a multi-user system with more than 2 users. Although the code design procedure described above for the 2-user system can be easily extended to systems with a greater number of users, we present in this paper a particularly elegant solution that we have found for the case where $N = 2^r$, where r is an arbitrary positive integer. As in the 2-user case, we start with a rate $1/N$ convolutional code, whose distance properties are known, say $\boldsymbol{G} = \left[\, g^1(D) \;\; g^2(D) \;\; \cdots \;\; g^N(D) \,\right]$, and assign it to source S_1. Thus,

$$\boldsymbol{G}_N^1(D) = \left[\, g^1(D) \;\; g^2(D) \;\; \cdots \;\; g^N(D) \,\right]. \tag{34}$$

We propose to assign to each of the remaining sources a cyclic shift of $\boldsymbol{G}_N^1(D)$. Thus, to source S_2 we assign transfer function matrix $\boldsymbol{G}_N^2(D)$ given by

$$\boldsymbol{G}_N^2(D) = \left[\, g^N(D) \;\; g^1(D) \;\; \cdots \;\; g^{N-1}(D) \,\right], \tag{35}$$

to source S_3 we assign transfer function matrix $\boldsymbol{G}_N^3(D)$ given by

$$\boldsymbol{G}_N^3(D) = \left[\, g^{N-1}(D) \;\; g^N(D) \;\; \cdots \;\; g^{N-2}(D) \,\right], \tag{36}$$

and so on. The composite $(N \times N)$ matrix is therefore described by

$$\boldsymbol{G}_N(D) = \begin{bmatrix} g^1(D) & g^2(D) & g^3(D) & \cdots & g^N(D) \\ g^N(D) & g^1(D) & g^2(D) & \cdots & g^{N-1}(D) \\ g^{N-1}(D) & g^N(D) & g^1(D) & \cdots & g^{N-2}(D) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g^2(D) & g^3(D) & g^4(D) & \cdots & g^1(D) \end{bmatrix}. \tag{37}$$

We now prove an interesting property of matrices having the structure of the composite matrix $\boldsymbol{G}_N(D)$ when N is a integral power of 2, *i.e.*, $N = 2^r$ where r is an arbitrary positive integer. We claim that the determinant of the composite matrix is given by

$$\Delta\left\{\boldsymbol{G}_N(D)\right\} = [g^1(D)]^N + [g^2(D)]^N + \cdots + [g^N(D)]^N. \tag{38}$$

In order to prove this statement, we need to prove two related properties of matrices and polynomials. The proofs are abbreviated here for lack of space.

Lemma 1 *For all N, such that N is an integral power of 2,* i.e., *$N = 2^r$, where r is a postive integer, the determinant of the $(N \times N)$ transfer function matrix defined as in (37) is given by*

$$\Delta\left\{G_N(D)\right\} = \left[g^1(D) + g^2(D) + \cdots + g^{N-1}(D) + g^N(D)\right]^N \tag{39}$$

Proof. Consider the $(N \times N)$ matrix $\boldsymbol{P}_r$ where $r = \log_2 N$, with the structure

$$\boldsymbol{P}_r = \begin{bmatrix} \boldsymbol{P}_{r-1} & \boldsymbol{P}_{r-1} \\ 0 & \boldsymbol{P}_{r-1} \end{bmatrix}, \tag{40}$$

and $\boldsymbol{P}_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. If the matrix $\boldsymbol{G}_N(D)$ is premultiplied and postmultipled by $\boldsymbol{P}_r$, the result is a triangular matrix, *i.e.*,

$$\boldsymbol{P}\boldsymbol{G}_N(D)\boldsymbol{P} = \begin{bmatrix} \Sigma & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ * & \Sigma & 0 & 0 & 0 & \cdots & 0 & 0 \\ * & * & \Sigma & 0 & 0 & \cdots & 0 & 0 \\ * & * & * & \Sigma & 0 & \cdots & 0 & 0 \\ * & * & * & * & \Sigma & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ * & * & * & * & * & \cdots & \Sigma & 0 \\ * & * & * & * & * & \cdots & * & \Sigma \end{bmatrix} \tag{41}$$

where $\Sigma = g^1(D) + g^2(D) + \cdots + g^{N-1}(D) + g^N(D)$, and $*$ represents a term which is inconsequential in this context. By the property of triangular matrices, the determinant of this matrix is given by

$$\Delta\left\{\boldsymbol{P}\boldsymbol{G}_N(D)\boldsymbol{P}\right\} = \left[g^1(D) + g^2(D) + \cdots + g^{N-1}(D) + g^N(D)\right]^N. \tag{42}$$

Lemma 2 *For all N, such that N is an integral power of 2,* i.e., *$N = 2^r$, where r is a postive integer,*

$$\left[g^1(D) + g^2(D) + \cdots + g^{N-1}(D) + g^N(D)\right]^N = [g^1(D)]^N + [g^2(D)]^N + \cdots + [g^N(D)]^N. \tag{43}$$

Outline of Proof. We group the N terms in the expression on the LHS into 2 subterms and express it as a binomial expansion. We then use the property of binomial coefficients and recognize that the coefficients of $g^i(D)$ are in $GF(2)$ to complete the proof.

Theorem 1 *The determinant of the $(N \times N)$ matrix $\boldsymbol{G}_N(D)$ whose structure is defined in (37), when $N = 2^r$ where r is an arbitrary positive integer is given by*

$$\Delta\left\{\boldsymbol{G}_N(D)\right\} = [g^1(D)]^N + [g^2(D)]^N + \cdots + [g^N(D)]^N. \tag{44}$$

Proof. Proof follows from Lemmas 1 and 2.

There are many ways of choosing $\{g^i(D)\}$ $(i = 1, \ldots, N)$, to ensure that $\Delta(D) = D^l$ for some $l \geq 0$. As an illustrative example, we can pick $\{g^i(D)\}$ $(i = 1, \ldots, N-1)$ to be the generator polynomials of the best $(N-1, 1, m)$ convolutional code published in the literature, and let

$$g^N(D) = g^1(D) + g^2(D) + \cdots + g^{N-1}(D) + D^p \text{ for some } p \geq 0. \tag{45}$$

With this choice of $\{g^i(D)\}$, we use (38) to show that

$$\Delta(D) = D^{Np}. \tag{46}$$

For example, for the 4−user multiple access system, *i.e.*, $N = 4$, we use the (3,1,2) convolutional code with the best d_{free} [8] and assign

$$\begin{aligned} g^1(D) &= 1 + D^2, && (47)\\ g^2(D) &= 1 + D + D^2, \quad \text{and} && (48)\\ g^3(D) &= 1 + D + D^2. && (49) \end{aligned}$$

We satisfy the sufficiency condition stated in (45) by choosing

$$g^4(D) = 1. \tag{50}$$

The composite (4×4) matrix is therefore described by

$$G_4(D) = \begin{bmatrix} 1+D^2 & 1+D+D^2 & 1+D+D^2 & 1 \\ 1 & 1+D^2 & 1+D+D^2 & 1+D+D^2 \\ 1+D+D^2 & 1 & 1+D^2 & 1+D+D^2 \\ 1+D+D^2 & 1+D+D^2 & 1 & 1+D^2 \end{bmatrix}. \tag{51}$$

With this choice of $G_4(D)$, $\Delta(D) = D^8$. The (4,1,2) convolutional code generated by any row of $G_4(D)$ has a free distance $d_{free} = 9$. Thus, when only one source is active, $d_{free} = 9$. As T, the number of active sources increases, d_{free} decreases monotonically, and when $T = 4$, $d_{free} = 1$. The relationship between T, the number of active sources and d_{free}, the free distance of the coding system is tabulated in Table 1. We must point out here that (45) is not a necessary condition for designing uniquely decodable and non-catastrophic coding systems, but only a sufficient one. Consequently, it is possible to design a composite transfer function matrix that generates uniquely decodable and non-catastrophic codes, without satisfying (45). For example, we have designed a 3-user system with

$$G_3(D) = \begin{bmatrix} 1+D^2 & 1+D+D^2 & 1 \\ 1 & 1+D^2 & 1+D+D^2 \\ 1+D+D^2 & 1 & 1+D^2 \end{bmatrix}. \tag{52}$$

This choice of $\{g^i(D)\}$ does not satisfy (45). However, the determinant of the composite matrix, $\Delta\{G_3(D)\} = D^6$, which makes the code generated by $G_3(D)$ and all its subsets uniquely decodable and non-catastrophic. The error-control capability of this system is also tabulated Table 1.

Finally we address the problem of determining $d_{free}(T)$, the free distance of the code generated by any T−row subset of $(N \times N)$ transfer function matrix. It is apparent that

$d_{free}(1) = d_{free}$, the free distance of the base code $G(D)$, and that $d_{free}(N) = 1$. We have used methods based on inspection and analytical techniques to determine the free distances $d_{free}(T), 1 < T < N$, for small values of m and N, and presented some of the results in Table 1. For larger values of m and N, we use a combination of numerical and analytical techniques to determine the free distances $d_{free}(T), 1 < T < N$.

V. CONCLUSIONS

In this paper we have described a new approach to designing an $N-$user multiple access system for a binary channel, based on convolutional codes. In the proposed system, the bandwidth left unused by the $N-T$ inactive users is used to provide error-control for the information transmitted by the T active users. We have described techniques that can be used to construct uniquely decodable and non-catastrophic codes for such systems. We have presented a detailed example of the design of a 2-user system, and have outlined an algorithm for designing systems with an arbitrarily large number of users, N, as long as N is an integral power of 2, *i.e.*, $N = 2^r$. We emphasize once again that (45) is not a necessary condition for designing uniquely decodable and non-catastrophic coding systems. Consequently, it is possible to design multi-user systems with an arbitrary number of users based on the concepts we have presented earlier. We have presented results that show how the error-control capability of the system varies with the number of active users.

VI. REFERENCES

[1] Peter Mathys. Codes for the T active users out of N multiple access channel. In *Proceedings of the 1988 Beijing International Workshop on Information Theory*, pages BI 3.1–7, Beijing, China, July 1988.

[2] Vivek Telang and Mark Herro. Error coding for the binary n-user modulo-q channel. In *Proc. of the International Symposium on Information Theory*, page 44, San Diego, California, Jan. 1990.

[3] James L. Massey and Michael K. Sain. Inverses of linear sequential circuits. *IEEE Trans. Comput.*, C-17:330–7, April 1968.

[4] G. David Forney. Convolutional codes I: Algebraic structure. *IEEE Trans. Inform. Theory*, IT-16:720–38, Nov. 1970.

[5] Roger Peterson and Daniel J. Costello, Jr. Binary convolutional codes for a multiple-access channel. *IEEE Trans. Inform. Theory*, IT-25:101–5, January 1979.

[6] James L. Massey, Daniel J. Costello, Jr., and Jørn Justesen. Polynomial weights and code constructions. *IEEE Trans. Inform. Theory*, IT-19:101–10, January 1973.

[7] Lalit Bahl and Frederick Jelinek. Rate 1/2 convolutional codes with complementary generators. *IEEE Trans. Inform. Theory*, IT-17:718–27, November 1971.

[8] Shu Lin and Daniel J. Costello, Jr. *Error Control Coding: Fundamentals and Applications.* Prentice-Hall, 1983.

A New Family of Algebraically Designed Optical Orthogonal Codes[*]

Zoran Kostić[‡], Svetislav V. Marić[†] and Edward L. Titlebaum[‡]

[‡] *Department of Electrical Engineering, University of Rochester, Rochester NY 14627.*
[†] *Winlab, Rutgers The State University of New Jersey, Piscataway NJ 08855.*

INTRODUCTION

In this paper we provide a new *algebraic construction* of $(n, w, \lambda_a, \lambda_c)$ Optical Orthogonal Codes (OOC's) for use in asynchronous Code Division Multiple Access (CDMA) Fiber-Optics Communication Systems with Optical Processing [1]. An optical orthogonal code, C, with parameters $n > 1, 1 \leq w \leq n, 0 \leq \lambda_a, \lambda_c \leq w$ is defined as a family of (0,1)-sequences of length n, with constant Hamming weight w and restricted values λ_a and λ_c of their out of phase auto-correlation and cross-correlation functions [2, 3]. For an arbitrary code sequence $u_m(i),\ i = 0, 1, ..., n-1$ (vector $\mathbf{u_m}$) from a code C , the following should be satisfied:

- The *sidelobes of the auto-correlation function* $r_{u_m,u_m}(\tau) = \sum_{i=0}^{n-1} u_m(i)u_m(i \oplus \tau)$, of $\mathbf{u_m}$, $\forall \mathrm{m}$ should be minimized. Thus $r_{u_m,u_m}(\tau) \leq \lambda_a$, for all integers $\tau \neq 0 (mod\, n)$, where $\oplus$ is addition $mod\ n$.
- The *peak of the auto-correlation function* $r_{u_m,u_m}(0) = \sum_{i=0}^{n-1} u_m(i)u_m(i)$, of $\mathbf{u_m}$, $\forall \mathrm{m}$ should be maximized.
- The *cross-correlation function* $r_{u_{m_1},u_{m_2}}(\tau) = \sum_{i=0}^{n-1} u_{m_1}(i)u_{m_2}(i \oplus \tau)$, of any two sequences $\mathbf{u_{m_1}}$ and $\mathbf{u_{m_2}}$ that belong to the code set should be minimized. Thus $r_{u_{m_1},u_{m_2}}(\tau) \leq \lambda_c$, for all integers τ.

Several methods for designing Optical Orthogonal Codes have been published by F. Chung et al. [2], H. Chung and Kumar [3] and Shaar and Davies [4].

QUADRATIC CONGRUENCE (QC) OPTICAL ORTHOGONAL CODES

A sequence $u_m(i), \quad i = 0, 1, 2, ..., n-1$, a member of a Quadratic Congruence OOC Code is constructed by

$$u_m(i) = \begin{cases} 1 & if\ \ y_m(k) + kp = i,\ \ i = 0, 1, ..., n-1 = p^2 - 1\ \ and\ \ k = \left\lfloor \frac{i}{p} \right\rfloor, \\ 0 & elsewhere \end{cases} \tag{1}$$

where p is an odd prime and the placement operator $y_m(k)$[†] is

$$y_m(k) \equiv m\frac{k(k+1)}{2} (mod\, p),\ \ 0 \leq k \leq p-1,\ \ 1 \leq m \leq p-1. \tag{2}$$

A graphical presentation of formulas 1 and 2 for $p = 5$ is given in the figure. Shaded squares represent ones, and empty squares represent zeros as elements of matrix and OOC sequence, respectively.

THEOREM : QC OOC's are $(p^2, p, 2, 4)$ codes for an odd prime p.

PROOF : The proof consists of three parts.

Part a) From equations 1 and 2 and the figure, there is a single 1 in each matrix column and each slot of a OOC sequence. Matrix columns map into OOC sequence slots. The number of matrix columns is equal to the number of sequence slots, and the number of matrix rows is equal to the number of sequence elements in a slot. Thus $w = p$ and $n = p^2$.

Part b) We show that the matrix $\mathbf{A}_m$ of size $p \times p$ with elements

---

[*] This work was supported in part by a grant from SDIO/IST and managed by the Office of Naval Research under contract N0014–86–K-0511

[†] In general, quadratic congruence codes with the placement operator $y_m(k) \equiv ak^2 + bk + c(mod\, p)$, can be used for design of QC OOC codes. There are some restrictions on values of a and b.

$$[a_m]_{k,y_m(k)} \equiv \begin{cases} 1 & if \;\; y_m(k) \equiv m\dfrac{k(k+1)}{2}(mod\ p) \\ 0 & otherwise \end{cases}, \qquad (3)$$

has at most one coincidence ($\rho_a \leq 1$) in its auto-coincidence matrix for any combination of vertical and horizontal shifts, considered cyclically or acyclically. Also, there are at most two coincidences ($\rho_c \leq 2$) in the cross-coincidence matrix of an arbitrary pair of matrices $\mathbf{A}_{m_1}, \mathbf{A}_{m_2}$ [5, 6]. A coincidence occurs when the difference function is equal to zero

$$\left(m_1\frac{(k+s_h)(k+s_h+1)}{2} + s_v - m_2\frac{k(k+1)}{2} \right) \equiv 0(mod\ p), \qquad (4)$$

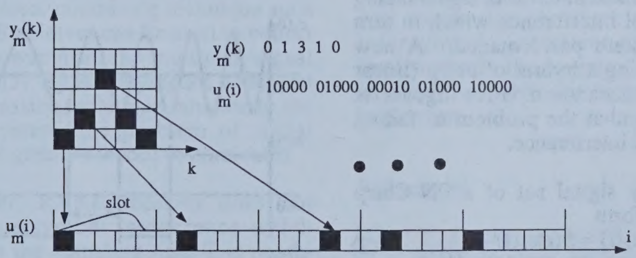
where s_h, s_v represent horizontal and vertical matrix shifts. For QC codes, the equation is quadratic and, by Lagranges theorem has a maximum of 2 noncongruent solutions ($\rho_c \leq 2$). When $m = m_1 = m_2$, the equation is linear, and has at most 1 noncongruent solution ($\rho_a \leq 1$).

Part c) We show that $\lambda_a \leq 2$ and $\lambda_c \leq 4$ by using ρ_a, ρ_c. From equations 1 and 2 and the figure, we see how the positions of one's in a sequence $u_m(i)$, $i = 0, 1, 2, ...n-1$ are determined by $y_m(k)$. The shift of τ between sequences $\mathbf{u}_\mathrm{i}$ and $\mathbf{u}_\mathrm{j}$ corresponds to either of the two following shift combinations between appropriate matrices:

- $\tau = qp + r, \quad 0 < r < p-1$
- $\tau = (q+1)p + l, \quad -(p-1) < l < 0$

where *q* represents the shift in the column of the matrix, and *r* and *l* represent shifts in the row of the matrix representing $y_m(k)$. To determine the number of coincidences between two shifted OOC sequences, the effect of both matrix shifts has to be considered. From *Part b)* of this proof we established that $\rho_c \leq 2$, which with the above claim maps into $\lambda_c = 2\rho_c = 4$, and $\rho_a \leq 1$ maps into $\lambda_a = 2\rho_a = 2$. Q.E.D.

Note that for every prime, *p*, we can have at most $M = p - 1$ different users in the system. The number of one's in each code sequence is *p* and hence the maximum of the auto-correlation function is equal to *p*.



REFERENCES

[1] P. Prucnal, M. Santoro, and T. Fan, "Spread spectrum fiber optic local area network using optical processing," IEEE Journal of Lightwave Technology, vol. LT-4, pp. 547–554, May 1986.

[2] F. R. K. Chung, J. A. Salehi, and V. K. Wei, "Optical orthogonal codes: Design, analysis and applications," IEEE Trans. Information Theory, vol. IT-35, pp. 595–604, May 1989.

[3] H. Chung and P. V. Kumar, "Optical orthogonal codes-new bounds and an optimal construction," IEEE Trans. Information Theory, vol. IT-36, pp. 866–872, July 1990.

[4] A. Shaar and P. Davies, "Prime sequences: Quasi-optimal sequences for or channel code division multiplexing," Electronic Letters, vol. 19, pp. 888–890, October 1983.

[5] E. Titlebaum and L. Sibul, "Time frequency hop signals part 2: Coding based upon quadratic congruences," IEEE Trans. Aerospace and Electronic Systems, pp. 494–501, July 1981.

[6] E. Titlebaum, J. Bellegarda, and S. Maric, "Ambiguity properties of quadratic congruential coding," IEEE Trans. Aerospace and Electronic Systems, to be published 1990.

PN-CHIRP SIGNALS FOR MULTIPLE-USER ANTI-MULTIPATH COMMUNICATION

C. H. Suhartanto and C. D. McGillem

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

Recent results in characterization of the indoor radio channel have made available information needed to design a high-performance, reliable communication system for use in environments such as factories, warehouses and office buildings.[1][2] The channel is modeled as a sum of M impulses representing M different paths that a transmitted signal goes through to reach an intended receiver. Each impulse has an associated path strength α_i and a propagation delay τ_i. That is,

$$h(t) = \sum_{i=1}^{M} Re\left\{\alpha_i \, \delta[t - \tau_i] \, exp\, \{j2\pi f_0 \tau_i\}\right\}. \tag{1}$$

Communication in such a channel is limited by the multipath time spread. This limitation is realized in terms of signal fading and intersymbol interference which in turn lowers the system performance. A new system employing a hybrid of chirp (linear FM) and pseudorandom (PN) signals is proposed to combat the problem of fading and intersymbol interference.

The binary signal set of a PN-Chirp system has the form

$$s_k(t) = b(t)c_k(t) \tag{2}$$

where the PN and the chirp signals are denoted by $b(t)$ and $c_k(t)$, respectively. The PN signal can be represented as

$$b(t) = \sum_{n=1}^{N-1} a_n f(t - nt_1) \tag{3}$$

where

$$f(t) = \begin{cases} 1, & 0 \le t \le t_1 \\ 0, & elsewhere \end{cases} \tag{4}$$

and

$$a_n = \pm\, 1. \tag{5}$$

The chirp signal has the form

$$c_k(t) = Re\left\{p(t)\, exp\left\{j\mu_k \frac{t^2}{2}\right\} exp\,\{j2\pi f_0 \tau_i\}\right\} \tag{6}$$

where

$$p(t) = \begin{cases} 1, & 0 \le t \le T = (N-1)t_1 \\ 0, & elsewhere \end{cases} \tag{7}$$

and μ_k, $k = 0,1$ are the dispersive slopes for which $\mu_0 = -\mu_1$. The block diagram of the PN-Chirp transmitter is shown in Figure 1 and the PN-Chirp signal is pictured in Figure 2.

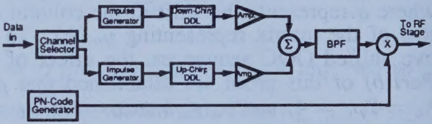


Figure 1. PN-Chirp Transmitter

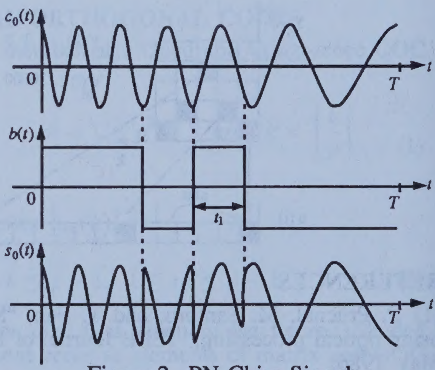


Figure 2. PN-Chirp Signal

At the receiver, two parallel chirp filters having dispersive slopes of μ_0 and μ_1 are used to detect the signal. Prior to the filtering process, the received signal is multiplied by a synchronized pseudonoise signature sequence to remove phase shifts of the PN modulation. A typical received signal in the presence of multiple (four) propagation paths is shown in Figure 3a. The multipath signal components are overlaping when they were received. To avoid such an overlap (signal fading) in a conventional communication system, the signal duration has to be much less than the multipath time spread, T_m. However, in an indoor environment, such a constraint would be too restrictive since the value of T_m is typically only a few hundred nanoseconds in such an environment. However, in a PN-Chirp system, this constraint can be relaxed to some extent. The output of the chirp filter, given the waveform of Figure 3a is the input, is shown in Figure 3b.

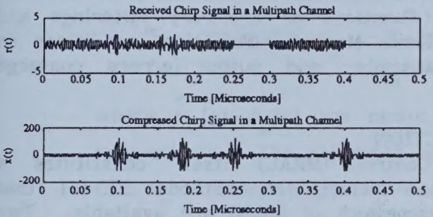


Figure 3. (a) Typical received signal in a multipath environment (b) Output of the chirp filter

Once the signal components are resolved, a linear combining technique such as the RAKE receiver can be used to collect the signal power from the multipath signal components.[3] By employing the wideband pulse compression technique inherent to the PN-Chirp system, the problem of signal fading can be greatly reduced or eliminated.

However, RAKE receiver does not eliminate intersymbol interference which occurs when the signaling time T is nearly equal or smaller than the coherence time of the channel, $(\Delta t)_c$. Two methods can be used to reduce the intersymbol interference problem. One is the inverse filter method also known as (adaptive) multipath channel equalization. The other is the echo cancelling technique that annihilates the multipath effects on the tranmitted signal.

The compressed chirp signal carries with it information about the channel parameters (i.e., the path strengths and the time delays of each path) together with the binary information signal from the transmitter. This channel parameter information can be extracted from the signal and be used to carry out the processing required by methods mentioned earlier.

The parameters measured from the received signal can be used to constuct an inverse channel in the following manner. The transfer function of the multipath channel modeled in time domain as in (1) is given by

$$H(f) = \sum_{i=1}^{M} \alpha_i \, exp\,\{-2\pi f \tau_i\}. \tag{8}$$

Given the pairs $(\hat{\alpha}_i, \hat{\tau}_i)$s, a filter to approximate the inverse of the multipath channel can be constructed. That is

$$G(f) \doteq \frac{1}{H(f)} \qquad |f - f_c| \le \frac{B}{2} \tag{9}$$

where B is the bandwidth of the transmitted signal.

The second method to make use of the measured channel parameters to reduce the problem of intersymbol interference can be realized in terms of an echo canceller. The impulse response of the echo canceller will have the form

$$g(t) = \hat{\alpha}_1 \delta\left[t - \hat{\tau}_1\right] - \sum_{i=2}^{M} \hat{\alpha}_i \, \delta\left[t - \hat{\tau}_i\right]. \tag{10}$$

Both of the system described in (9) and (10) will eventually eliminate the problem of intersymbol interference if properly implemented. The inverse filter will provide an improved SNR but is more difficult to implement.

References:

[1] T. S. Rappaport, "Characterization of UHF Factory Multipath Channels", Ph.D. Dissertation, School of Electrical Engineering, Purdue University, West Lafayette, Indiana, June 1988.

[2] P. Yegani, "Multiple Access Radio Communications in the Factory Environment", Ph.D. Dissertation, School of Electrical Engineering, Purdue University, West Lafayette, Indiana, December 1989.

[3] R. Price, P. E. Green, "A Communication Technique for Multipath Channels", Prodeedings of the IRE, Vol. 46, March 1958, pp. 555-570.

A Modified Output Error Method for Model Reference Adaptive Control

Yannis N. M. Papadakis and **Stelios C. A. Thomopoulos**[1]

Decision and Control Systems Laboratory
Department of Electrical & Computer Engineering
The Pennsylvania State University
University Park, PA 16802

ABSTRACT A Modified Output Error Method (MOEM) for Model Reference Adaptive Control is introduced. By properly defining the augmented error and the regressors, the global asymptotic stability of the resulting closed loop system is guaranteed without overparameterization. With the new regressors, the output error satisfies a constructable stable filtered equation. This method does not involve any strictly positive real transfer functions or arbitrary filterings and retains the intuitive appeal of the Output Error Method. Sufficient conditions on the reference input under which the parameter and output errors converge exponentially to zero are also derived.

INTRODUCTION

Although Model Reference Adaptive Control (MRAC) itself constitutes a particular approach to adaptive control, a variety of methods [1], [6] that achieve global asymptotic stability in the framework of MRAC are available. Two of these methods are of particular interest, namely the Output Error Method (OEM) and the Input Error Method (IEM). The OEM, described in [6]÷[8], has several variations. For example in [7], an additional parameter is introduced so that a difficulty in using the filtered error equation can be bypassed. However, this results in an overparameterization of the identifier. In [8], the previous approach is refined by introducing Strictly Positive Real (SPR) transfer functions. This has been avoided in the IEM [1]÷[3] by the introduction of a properly constructed signal (input error). For comparisons between the OEM and the IEM the reader is referred to [1], [2] and [10]. In the IEM, an a priori known bound on the high frequency gain k_p of the transfer function of the plant is necessary, so that all signals in the closed loop be bounded. A similar requirement for k_p holds for the MOEM and the uniform boundeness of the parameter error is guaranteed by the use of the gradient algorithm with projection which is also used in IEM. As in the IEM, the MOEM leads to a linear error equation for which other identification algorithms, i.e. least squares, are applicable and constitute an advantageous alternative to the gradient algorithm.

In the following analysis, the asymptotic stability and parameter convergence properties of the MOEM for the n-th order Single Input - Single Output (SISO) plant are established. Simulations of a second order plant under persistent and non-persistent excitation have been obtained using the MOEM and for reference reasons, using the IEM as well.

MRAC VIA MOEM : ANALYSIS

The notation used is that of [1], [2]. The assumptions on the plant, model and reference input (command) are as follows :

(A1) *Plant Assumptions*

The plant is a SISO Linear Time Invariant (LTI) system, described by a transfer function

---

[1] This work is partially supported by SDIO/IST and managed by the Office of Naval Research under Contract N0014-86-K-0515.

$$\frac{\hat{y}_p(s)}{\hat{u}(s)} = \hat{P}(s) = k_p \frac{\hat{n}_p(s)}{\hat{d}_p(s)} \tag{1.1}$$

where $\hat{n}_p(s)$, $\hat{d}_p(s)$ are monic, coprime polynomials of degree m and n respectively. The plant is strictly proper, and minimum phase. Without loss of generality we may assume that $k_p > 0$. A positive lower bound k_{pmin} on k_p is assumed to be known.

(A2) *Reference Model Assumptions*

The reference model is described by

$$\frac{\hat{y}_m(s)}{\hat{r}(s)} = \hat{M}(s) = k_m \frac{\hat{n}_m(s)}{\hat{d}_m(s)} \tag{1.2}$$

where $\hat{n}_m(s)$, $\hat{d}_m(s)$ are monic, coprime polynomials of degree m and n respectively. The reference model is stable, minimum phase and $k_m > 0$ (i.e. k_m is a known positive constant).

(A3) *Reference Input Assumptions*

The reference input $r(\cdot)$ is piecewise continuous and bounded on $\mathbb{R}^+$.

Controller Structure

The controller structure of Narendra, Lin and Valavani [8] is considered. The characteristic polynomial of the non-minimal observer is given by

$$\hat{\lambda}(s) = \hat{\lambda}_0(s)\, \hat{n}_m(s) \tag{1.3}$$

where $\hat{\lambda}_0(s)$ is an arbitrary monic Hurwitz polynomial of degree n−m−1. The pair $\Lambda \in \mathbb{R}^{n-1 \mathrm{x} n-1}$, $b_\lambda \in \mathbb{R}^{n-1}$ is chosen so that (Λ, b_λ) is in controllable canonical form and $\det(sI-\Lambda)=\hat{\lambda}(s)$. The adaptive observer is then defined by

$$\dot{w}_1 = \Lambda w_1 + b_\lambda u \qquad \dot{w}_2 = \Lambda w_2 + b_\lambda y_p \tag{1.4}$$

where the states w_1, $w_2 \in \mathbb{R}^{n-1}$, and the initial conditions $w_1(0)$, $w_2(0)$ are arbitrary. The control u is given by

$$u = c_0 r + c^T w_1 + d_0 y_p + d^T w_2 := c_0 r + \bar{\theta}^T \bar{w} := \theta^T w \tag{1.5}$$

where

$$\theta^T := (c_0, \bar{\theta}^T) := (c_0, c^T, d_0, d^T) \in \mathbb{R}^{1+(n-1)+1+(n-1)} = \mathbb{R}^{2n} \tag{1.6}$$

is the vector of *controlled parameters*, and

$$w^T := (r, \bar{w}^T) := (r, w_1^T, y_p, w_2^T) \in \mathbb{R}^{1+(n-1)+1+(n-1)} = \mathbb{R}^{2n} \tag{1.7}$$

is the regressor vector. The vector

$$\theta^{*T} := (c_0^*, \bar{\theta}^{*T}) := (c_0^*, c^{*T}, d_0^*, d^{*T}) \in \mathbb{R}^{1+(n-1)+1+(n-1)} = \mathbb{R}^{2n} \tag{1.8}$$

is the vector of the *nominal* controller parameters that achieves matching of the transfer function $r \to y_p$ to the model transfer function $\hat{M}$ [1], [6]. Notice that the first element, c_0^*, of θ^* is given by $c_0^* = k_m/k_p$.

The parameter error vectors are defined by

$$\phi := \theta - \theta^{*} \in \mathbb{R}^{2n} \qquad \bar{\phi} := \bar{\theta} - \bar{\theta}^{*} \in \mathbb{R}^{2n-1} \tag{1.9}$$

Identifier Structure

The output error e(t) is defined by

$$e := y_p - y_m \tag{2.1}$$

A fundamental identity similar to the one that holds for the IEM [1] is shown. The difference lies in the absence of the (n-m)th order arbitrary stable minimum phase filtering $\hat{L}(s)$ which is necessary for the implementation of the IEM.

Fundamental Identity

Given the assumptions (A1) and (A2) , the controller configuration of Figure 1 and the definitions of θ^{*} and $\bar{w}$ by (1.7) and (1.8), for any piecewise continuous $u \in L_{\infty e}$[2] and for arbitrary initial conditions in the realizations of the transfer functions, the following equation is satisfied.

$$y_p = \hat{M}\left\{\begin{bmatrix} 1/c_0^{*} & -\bar{\theta}^{*T}/c_0^{*} \end{bmatrix} \begin{bmatrix} u \\ \bar{w} \end{bmatrix}\right\} + \varepsilon(t) \tag{2.2}$$

where the term ε(t) is due to the initial conditions and is exponentially decaying.

The proof of (2.2) involves standard algebraic manipulations and is omitted. The exponentially decreasing term ε(t) plays no role in the convergence and stability of the adaptive method in MRAC [6] and is omitted in all subsequent derivations.

The new regressor and nominal parameter vector are defined by

$$\tilde{\theta}^{*T} := [1/c_0^{*} \quad \bar{\theta}^{*T}/c_0^{*}] \in \mathbb{R}^{2n}, \quad \tilde{w}^{T} := [-u \quad \bar{w}^{T}] \in \mathbb{R}^{1+(2n-1)} = \mathbb{R}^{2n} \tag{2.6}$$

and therefore eq. (2.2) is written as

$$y_p = -\hat{M}(\tilde{\theta}^{*T}\tilde{w}) \tag{2.7}$$

Notice that since $c_0^{*} = k_m / k_p$ and both k_m, k_p are non-zero, $\tilde{\theta}^{*}$ is well defined. Equation (2.7) is important since it holds for any piecewise continuous u. In order to form the error equation, we solve (1.5) with respect to the reference input r and obtain

$$r = (1/c_0)u - (1/c_0)\bar{\theta}^{T}\bar{w} \tag{2.8}$$

where we assumed $c_0(t) \neq 0 \;\; \forall\, t \geq 0$. This assumption will be justified in the stability proof of the method. From (1.2) and (2.8), y_m may be expressed as

$$y_m = -\hat{M}(\tilde{\theta}^{T}\tilde{w}) \tag{2.9}$$

where the new parameter vector is defined by

$$\tilde{\theta}^{T} := [1/c_0 \quad \bar{\theta}^{T}/c_0] \in \mathbb{R}^{2n} \tag{2.10}$$

Subtracting (2.9) from (2.7) and using (2.1), we obtain **the MOEM error equation**

$$e = \hat{M}(\tilde{\phi}^{T}\tilde{w}) \tag{2.11}$$

where the new parameter error vector is defined by

$$\tilde{\phi} := \tilde{\theta} - \tilde{\theta}^{*} = \begin{bmatrix} (1/c_0)-(1/c_0^{*}) & (\bar{\theta}^{T}/c_0)-(\bar{\theta}^{*T}/c_0^{*}) \end{bmatrix}^{T} \in \mathbb{R}^{2n} \tag{2.12}$$

---

2. $u \in L_{\infty e}$ iff $u_s \in L_{\infty}$ where $u_s(t) := u(t)$ for $t \leq s$ and $u_s(t) := 0 \;\; \forall\, t > s$.

Remarks

1. By properly defining the regressors and the parameter error vectors, a constructable error equation is obtained which has the form of a linear equation on the parameters but involves the dynamics of model contained in $\hat{M}$.
2. Although equation (2.11) requires that u is always equal to the computed value $\theta^T w$, it has been proved that only one part of this equation, i.e. (2.9), requires that. Input saturation phenomena can be introduced by modifying the control law (1.7) so that it saturates when $|u|$ exceeds some constant value. The analysis and performance of the MOEM under this realistic constraint is being investigated.
3. Equation (2.11) and its properties have been extensively studied in [9] in which the violation of SPR-ness of $\hat{M}$ in the high frequencies is quantified through a cumbersome analysis and is related to bounds on the regressor and its frequency content. In the OEM, the output error equation is given by ([1] p. 118, eq. 3.3.21)

$$e = (1/c_0^*)\ \hat{M}(\phi^T w)$$

and since c_0^* is unknown, $\hat{M}$ should be SPR unless some modification is made. This is not the case with the MOEM output error equation (2.6), which may be used to generate the adaptation laws determining the evolution of the parameter vector θ, without any SPR requirements on $\hat{M}$.
4. A similar approach may be used for the identification problem and is addressed in [11].
5. The equation $\tilde{\phi} = 0$ is satisfied **if and only if** $\theta = \theta^*$.

The augmented error [5] is defined by

$$\tilde{e} := \tilde{\phi}^T\hat{M}(\tilde{w}) = e + \tilde{\theta}^T\hat{M}(\tilde{w}) - \hat{M}(\tilde{\theta}^T\tilde{w}) \tag{2.13}$$

$$= e + \tilde{\phi}^T\hat{M}(\tilde{w}) - \hat{M}(\tilde{\phi}^T\tilde{w}) := e + e_1$$

where e_1 is usually called the auxiliary error and $\hat{M}(\tilde{w})$ is understood as application of $\hat{M}$ on each element of $\tilde{w}$, resulting in a 2n-th order vector. The signal $\tilde{e}$ is constructable since it is given as a function of known signals.

Using (2.13) and the normalized gradient algorithm with projection, we obtain the MOEM adaptation law

Adaptation Law *(Normalized Gradient Alg. with Projection)*

$$\dot{\tilde{\phi}} = \dot{\tilde{\theta}} = -\ g\ \frac{\tilde{e}\ \hat{M}(\tilde{w})}{1 + \gamma\ \hat{M}(\tilde{w}^T)\hat{M}(\tilde{w})} \qquad g,\gamma > 0 \tag{2.14}$$

Projection Scheme

if $c_0 = c_{0max}$ $(:= k_m/k_{pmin})$ and $\dot{c}_0 > 0$ **then** set $\dot{c}_0 = 0$.

We obtain analytic expressions for the elements of $\tilde{\theta}$ via (2.14). From (2.10) and considering that $\tilde{\theta}$ and $\tilde{w}$ may be written as

$$\tilde{\theta}^T = \begin{bmatrix} 1/c_0 & \bar{\theta}_1/c_0 & \bar{\theta}_2/c_0 & \cdots & \bar{\theta}_{2n-1}/c_0 \end{bmatrix} \in \mathbb{R}^{2n}$$

$$\tilde{w}^T = \begin{bmatrix} \tilde{w}_0 & \tilde{w}_1 & \cdots & \tilde{w}_{2n-1} \end{bmatrix} \in \mathbb{R}^{2n} \tag{2.15}$$

after differentiation and use of (2.14), we obtain the differential equations

$$\dot{c}_0 = g \frac{\tilde{e}\ \hat{M}(\tilde{w}_0)}{1 + \gamma\ \hat{M}(\tilde{w}^T)\hat{M}(\tilde{w})} (c_0)^2$$

$$\frac{d}{dt}\left(\frac{\bar{\theta}_i}{c_0}\right) = -\ g \frac{\tilde{e}\ \hat{M}(\tilde{w}_i)}{1 + \gamma\ \hat{M}(\tilde{w}^T)\hat{M}(\tilde{w})} \qquad 1 \le i \le 2n-1 \tag{2.16}$$

or equivalently, the integral equations

$$c_0(t) = \frac{1}{-\displaystyle\int_0^t g \frac{\tilde{e}\ \hat{M}(\tilde{w}_0)}{1 + \gamma\ \hat{M}(\tilde{w}^T)\hat{M}(\tilde{w})} d\tau + 1/c_0(0)}$$

$$\bar{\theta}_i(t) = \int_0^t \left[g \frac{c_0 \tilde{e}}{1 + \gamma\ \hat{M}(\tilde{w}^T)\hat{M}(\tilde{w})}\right]\left[\bar{\theta}_i \hat{M}(\tilde{w}_0) - \hat{M}(\tilde{w}_i)\right] d\tau + \bar{\theta}_i(0) \qquad 1 \le i \le 2n-1 \tag{2.17}$$

Global Stability

Theorem 1

Under the assumptions (A1), (A2), (A3) and the adaptation law (2.14), the following holds true for arbitrary initial conditions with $0 < c_0(0) \le c_{0max}$

(a) all signals in the closed loop are bounded

(b) $e(t) \in L_2$ and $\lim_{t\to\infty} e(t) \to 0$, i.e. the output error converges asymptotically to zero.

Proof

We follow a similar approach to that of [6], used for the proof of the asymptotic stability for the OEM. The initial conditions terms are omitted since they do not affect the stability results.

Step 1.

By a well known theorem (Thm. 2.4.2 in [1], p. 64), the use of the normalized gradient algorithm (2.14) associated with the augmented error equation (2.13) assures that:

$$\frac{\tilde{e}}{\sqrt{1 + \gamma\hat{M}(\tilde{w}^T)\hat{M}(\tilde{w})}} \in L_2 \cap L_\infty,\ \tilde{\phi} \in L_\infty,\ \dot{\tilde{\phi}} \in L_2 \cap L_\infty,\ \tilde{\theta} \in L_\infty$$

$$\tilde{e} = \beta(t) \sqrt{1 + \gamma\hat{M}(\tilde{w}^T)\hat{M}(\tilde{w})},\quad \beta(t) \in L_2 \cap L_\infty \tag{2.18}$$

From (2.12) and (2.18), the boundeness of $\tilde{\phi}$ implies that $1/c_0$ is always bounded which in turn implies that $c_0 \ne 0\ \forall t \ge 0$. Furthermore, the projection scheme in eq. (2.14) guarantees that c_0 is always bounded so that $1/c_0$ is always bounded away from zero. Therefore, the L_∞ results for $\tilde{\theta}$ and $\tilde{\phi}$ are also true for the resulting θ and ϕ.

Step 2.

The closed loop system consisting of the plant, the reference model and the controller can be expressed in a linear time varying state form as follows

$$\dot{x} = \left[A_{mn} + b_{mn}\bar{\phi}^T(t)C\right] x(t) + b_{mn}c_0(t)r(t) \tag{2.19}$$

where A_{mn} and C are constant matrices, b_{mn} is a constant vector and $x \in \mathbb{R}^{3n-2}$ is the state vector of the closed loop system. Since the parameter error is uniformly bounded, eq. (2.19) can be considered as a linear time-varying differential equation with bounded coefficients[3]. It follows that $\|x(t)\|$ grows at most exponentially.

Step 3.

Assuming that all signals in the loop grow in an unbounded fashion, and comparing their growth rates we obtain that

$$\sup_{\tau\le t}\|y_p(\tau)\| \sim \sup_{\tau\le t}\|w_2(\tau)\| \sim \sup_{\tau\le t}\|\bar{w}(\tau)\| \sim \sup_{\tau\le t}\|w\|$$

$$|u(t)|, \|w_1(t)\| = O\left[\sup_{\tau\le t}\|\bar{w}(\tau)\|\right] \tag{2.20}$$

where the same notation as in [6] was introduced and is explained in the Appendix. From the definition of $\tilde{w}$, we have that

$$\tilde{w} = \begin{bmatrix} -c_0 & -\bar{\theta}^T \\ 0 & I \end{bmatrix} \begin{bmatrix} r \\ \bar{w} \end{bmatrix} := G(t)\ w \tag{2.21}$$

where $G(t) \in \mathbb{R}^{2n\times2n}$ is a bounded nonsingular operator. Therefore we conclude that $\tilde{w} \sim w$, i.e. $\tilde{w}$ and w grow at the same rate.

Step 4.

From (2.13), the auxiliary error e_1 is given by

$$e_1 = \tilde{\phi}^T\hat{M}(\tilde{w}) - \hat{M}(\tilde{\phi}^T\tilde{w}) \tag{2.23}$$

and since $\dot{\tilde{\phi}} \in L_2$ from the "swapping lemma" [6], e_1 satisfies the equation

$$e_1 = o\left[\sup_{\tau\le t}\|\tilde{w}(\tau)\|\right] \tag{2.24}$$

The output of the plant can be expressed as

$$y_p(t) = y_m(t)+\tilde{e}-e_1 = y_m(t)+\beta(t)\sqrt{1 + \gamma\hat{M}(\tilde{w}^T)\hat{M}(\tilde{w})} + o\left[\sup_{\tau\le t}\|\tilde{w}(\tau)\|\right] \tag{2.25}$$

and by using (1.4) we have that

$$\|w_2(t)\| \le \|(sI-\Lambda)y_m\| + o\left[\sup_{\tau\le t}\|\hat{M}(\tilde{w})\|\right] + o\left[\sup_{\tau\le t}\|\tilde{w}(\tau)\|\right] \tag{2.26}$$

and therefore,

$$\|w_2(t)\| = o\left[\sup_{\tau\le t}\|\tilde{w}(\tau)\|\right] \tag{2.26}$$

which contradicts the conclusions of Step 3, i.e. $w_2 \sim \tilde{w}$. Hence all signals in the feedback loop are uniformly bounded.

Step 5.

From Step 4, (2.13) and (2.18), we conclude that $\dot{\tilde{e}}$ is bounded. Since $\tilde{e}$ is in

---

[3] **For a detailed analysis on the existence of solutions of (2.14), (2.19) see [12].**

L_2 and its derivative is bounded, it converges asymptotically to zero. From (2.24), we conclude that the auxiliary error e_1 tends asymptotically to zero and thus the output error $e(t)$ is in L_2 and converges to zero asymptotically. ■

In the following equations, the subscript m implies that θ is equal to θ^*. The expression for $\tilde{w}_m$ is given by

$$\tilde{w}_m = \begin{bmatrix} -\hat{P}^{-1}\hat{M} \\ (sI-\Lambda)^{-1}b_\lambda \hat{P}^{-1}\hat{M} \\ \hat{M} \\ (sI-\Lambda)^{-1}b_\lambda \hat{M} \end{bmatrix}(r) \tag{2.27}$$

Since

$$u = \hat{P}^{-1}[e + \hat{M}(r)] = \hat{P}^{-1}(e) + \hat{P}^{-1}\hat{M}(r) \tag{2.28}$$

it follows that

$$\tilde{w} = \begin{bmatrix} -1 \\ (sI-\Lambda)^{-1}b_\lambda \\ \hat{P} \\ (sI-\Lambda)^{-1}b_\lambda \hat{P} \end{bmatrix}(u) = \tilde{w}_m + \begin{bmatrix} -\hat{P}^{-1} \\ (sI-\Lambda)^{-1}b_\lambda \hat{P}^{-1} \\ 1 \\ (sI-\Lambda)^{-1}b_\lambda \end{bmatrix}(e) := \tilde{w}_m + \hat{A}(e) \tag{2.29}$$

where each block of $\hat{A}$ is appropriately dimensioned. After applying the stable operator $\hat{M}$ on both sides of (2.29) we have that

$$\hat{M}(\tilde{w}) = \hat{M}(\tilde{w}_m) + \hat{A}\hat{M}(e) = \hat{M}(\tilde{w}_m) + \begin{bmatrix} -\hat{P}^{-1}\hat{M} \\ (sI-\Lambda)^{-1}b_\lambda \hat{P}^{-1}\hat{M} \\ \hat{M} \\ (sI-\Lambda)^{-1}b_\lambda \hat{M} \end{bmatrix}(e) \tag{2.30}$$

The following theorem provides sufficient conditions for the exponential convergence of the parameter and output errors to zero.

Theorem 2

Under the assumptions (A1), (A2), (A3) and the adaptation law (2.14), and for arbitrary initial conditions with $0 < c_0(0) < c_{0max}$ we have that

$$\hat{M}(\tilde{w}) - \hat{M}(\tilde{w}_m) \in L_2$$

Proof

From Theorem 1, it follows that $\tilde{w}$ and therefore $\hat{M}(\tilde{w})$, are bounded. Since $e \in L_2$, using (2.30) and a well known property of asymptotically stable linear time invariant systems, the conclusion of the theorem follows. ■

The following theorem provides sufficient conditions for exponential convergence of the parameter and output errors to zero.

Theorem 3

Consider the assumptions (A1), (A2), (A3), the adaptation law (2.14) and arbitrary initial conditions with $0 < c_0(0) < c_{0max}$. *If* the spectral density of

$r(\cdot)$ contains at least 2n points *then* the parameter error $\tilde{\phi}(t)$ converges exponentially to zero.

Proof

The proof is based on well known theorems from generalized harmonic analysis [4] and therefore we only give an outline of the full proof. From (2.13) and the associated gradient law (2.14), it suffices to obtain conditions such that $\hat{M}(\tilde{w})$ is persistently exciting (PE), i.e. that there exist constants a, b and T > 0 such that

$$bI \ge \int_t^{t+T} \hat{M}(\tilde{w})\hat{M}(\tilde{w}^T)d\tau \ge aI \quad \text{for all } t\ge 0.$$

From (2.30) and Theorems 1 and 2, a sufficient condition for $\hat{M}(\tilde{w})$ to be PE is that $\hat{M}(\tilde{w}_m)$ is PE and finally that $\tilde{w}_m$ is PE, since $\hat{M}$ is an asymptotically stable LTI system. The requirements on r so that $\tilde{w}_m$ is PE are obtained by (2.27) and a theorem from generalized harmonic analysis adapted to the MRAC framework (e.g. see Theorem 2.7.3. in [1] and similar theorem in [4]). ■

Remarks

1. We assume that $r(\cdot)$ is such that its autocovariance exists, i.e. r is stationary. This is necessary so that the conclusion of Theorem 3 is established.
2. In the case of persistent excitation and since the regressor $\tilde{w}$ is bounded, the error equation (2.11) gives that the output error converges to zero exponentially.
3. The fact $\tilde{\phi} \to 0$ exponentially implies that $\phi \to 0$ exponentially.

SIMULATIONS

MRAC simulations for an unstable second order plant are obtained. The effect of persistent excitation in parameter convergence is demonstrated in the results, for which we have used persistent and non-persistent excitation. The plant and model transfer functions are given by $\hat{P}(s) = 5(s+1)/(s^2+s-6)$ and $\hat{M}(s) = 5(S+2)/(s^2+2s+10)$. The simulations shown in Figure 1 (a)÷(b) were obtained with r(t)=5, i.e. a non-PE input for a second order system. Although, as expected [4], the output error converges to zero in both the IEM and the MOEM, the parameter vector saturates to a value other than θ^*. In Figure 1 (c)÷(d), the input r is changed to r(t)=10sin(t)+5sin(5t) and is sufficiently rich of order 4, i.e. its spectral density has 4 spectral lines. Simulation results verify that θ converges to the nominal θ^* as shown in Figure 1 (d).

CONCLUSION

The Modified Output Error Method (MOEM) that is introduced in this paper, though based on the Output Error Method (OEM), avoids some drawbacks of the latter by properly defining the parameter and regressor vectors while retaining the intuitively appealing framework of the OEM. Overparameterization of the identifier is avoided without the introduction of any arbitrary stable filterings. The derived error equation is a constructable stable linear differential equation and leads to adaptation laws that guarantee asymptotic stability without strictly positive realness assumptions on any transfer function of the subsystems of the closed loop. Furthermore, when the reference input is sufficiently rich of order 2n, the parameter and output errors converge exponentially to zero.

REFERENCES

[1] S. Sastry and M. Bodson, Adaptive control, Stability Convergence, and Robustness, Prentice Hall, New Jersey, 1989.

[2] M. Bodson and S. Sastry, "Input Error versus Output Error Model Reference Adaptive Control," Proc. of the American Control Conference, pp. 224-229, Minneapolis, Minnesota, 1987.

[3] M. Bodson "Effects of the Choice of the Error Equation on the Robustness Properties of Adaptive Control Systems," International Journal of Adaptive Control and Signal Processing", Vol. 2, pp. 249-257, 1988.

[4] S. Boyd and S. Sastry, "On Parameter Convergence in Adaptive Control," Systems and Control Letters, no. 3, pp. 311-319, 1983.

[5] R. V. Monopoli, "Model Reference Adaptive Control with an Augmented Control Signal," IEEE Trans. on Automatic Control, Vol. AC-19, no. 5, pp. 474-484, 1974.

[6] K. S. Narendra and A. M. Annaswamy, Stable Adaptive Systems, Prentice Hall, New Jersey, 1989.

[7] K. S. Narrendra and Y. H. Lin, "Design of Stable Model Reference Adaptive Controllers," Applications of Adaptive Control, pp. 69-130, Academic Press, New York, 1980.

[8] K. S. Narendra and L. Valavani, "Stable Adaptive Controller Design - Part II: Proof of Stability", IEEE Transactions in Automatic Control, Vol. AC-25, pp. 440-448, June 1980.

[9] B. D. O. Anderson, R. R. Bitmead, C. R. Johnson, P. V. Kokotovic, R. L. Kosut, I. M. Y. Mareels, L. Praly and B. D. Riedle, Stability of Adaptive Systems, Passivity and Averaging Analysis, MIT Press, Cambridge, Massachusetts, 1986.

[10] G. C. Goodwin and D. Q. Mayne, "A parameter Perspective of Continuous Time Model Reference Adaptive Control," Automatica, Vol. 23, No. 1, pp. 57-70, 1987.

[11] S. C. A. Thomopoulos and Y. N. M. Papadakis, "A Duality Approach to Model Reference Adaptive Identification and Control," submitted to the 1991 American Control Conference.

[12] S. C. A. Thomopoulos and Y. N. M. Papadakis, "On the Existence of Solutions in Adaptive Control," Proceedings of the 29th Control and Decision Conference, Honolulu, Hawaii, December 1990, to appear.

APPENDIX

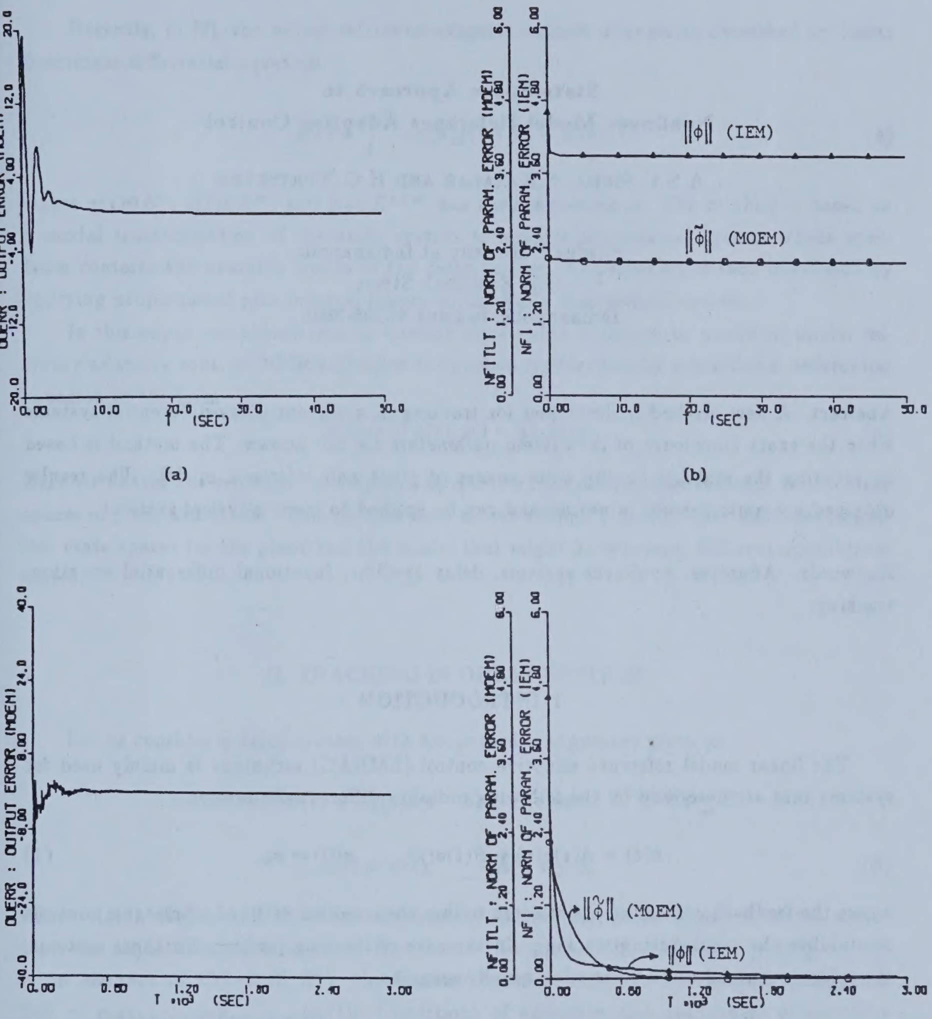
The following definitions [6] apply to all real piecewise continuous signals defined on $[0,\infty)$ which have bounded discontinuities.

Definition A1. We denote $y(t) = O[x(t)]$ if there exist positive constants M_1, M_2 and $t_0 \in \mathbb{R}^+$ such that $|y(t)| \leq M_1|x(t)| + M_2 \ \forall \ t \geq t_0$.

Definition A2. We denote $y(t) = o[x(t)]$ if there exists a piecewise continuous function $\beta(t)$ with bounded discontinuities and $t_0 \in \mathbb{R}^+$ such that $|y(t)| \leq \beta(t)x(t)$ $\forall \ t \geq t_0$ and $\lim_{t\to\infty} \beta(t) = 0$.

Definition A3. If $y(t) = O[x(t)]$ and $x(t) = O[y(t)]$ then x and y are said to be equivalent and denoted by $x(t) \sim y(t)$.

Remark : If $x(t) \sim y(t)$ then x and y are said to grow at the same rate.



(a) (b) (c) (d)

FIGURE 1 : In (a),(b) r(t)=5. (a) The output error goes to zero; (b) the parameter error does not converge to zero. In (c),(d) r(t)=10sin(t)+5sin(t). (c) The output error goes to zero; (d) the parameter error goes to zero.

State Space Approach to Nonlinear Model Reference Adaptive Control

A.S.C.Sinha, S.Kayalar and H.O.Yurtseven

Purdue University at Indianapolis
1201 East 38th Street
Indianapolis, Indiana 46205-2868

Abstract. A new method is developed for tracking in nonlinear delay-differential systems when the exact knowledge of the system parameters are not known. The method is based on securing the stability in the state spaces of plant and reference model. The results obtained are quite general in nature and can be applied to many physical systems.

Keywords. Adaptive, nonlinear systems, delay systems, functional differential equation, tracking.

I. INTRODUCTION

The linear model reference adaptive control (LMRAC) technique is mainly used for systems that are described by the following ordinary differential equation

$$\dot{x}(t) = A(t)x(t) + B(t)u(t), \qquad x(0) = x_0, \tag{1}$$

where the feedback control $u(\cdot)$ is selected within the compact set $\mathcal{U}$ of admissable controls to stabilize the plant during tracking. In the case of tracking problem for these systems, the reference model to be followed is usually given as

$$\dot{x}_r(t) = A_r x_r(t) + B_r u(t), \qquad x_r(0) = x_{r0}, \tag{2}$$

where A_r and B_r are constant matrices. Therefore, the state error $x_e(t) = x(t) - x_r(t)$ is governed by the following differential equation

$$\dot{x}_e(t) = A_r x_e(t) + A_e(t)x(t) + B_e(t)u(t), \qquad x_e(0) = x_0 - x_{r0}, \tag{3}$$

where $A_e(t) = A(t) - A_r$ and $B_e(t) = B(t) - B_r$. We then say that the plant follows the model if $x_e(t) \to 0$, $A_e(t) \to 0$, and $B_e(t) \to 0$ as $t \to \infty$. The literature on LMRAC systems described by ordinary differential equation is vast, see for review reference [1].

Recently, in [2], the model reference adaptive control of systems described by linear functional differential equation

$$\dot{x}(t) = \int_{-\tau}^{0} d\alpha(\theta)x(t+\theta) + B_0u(t) \tag{4}$$

where $x(t)\in E^n$, $u(t)\in E^m$, and $B_0\in E^{n\times m}$ has been investigated. The method is based on a modal transformation of the delay system to a finite dimensional system whose spectrum contains the unstable modes of the delay system. The tracking is then developed by applying proportional plus integral theory to the finite dimensional system.

In this paper our objective is to develop state space approach to nonlinear model reference adaptive control (NMRAC) when the system is described by a functional differential equation of the form

$$\dot{x}(t) = f(x(t), x(t-h), u(t)) \tag{5}$$

Here we obtain the required convergence by securing the asymptotic stability in the state spaces of plant and model. This method then allows arbitrary nonlinearity and noncompatible state spaces for the plant and the model that might have several different equilibrium states.

II. TRACKING IN DELAY SYSTEMS

Let us consider a delay system with one retarded argument given as

$$\dot{x}(t) = A_0(t)x(t) + A_1(t)x(t-h) + B(t)u(t), \qquad 0 < t \leq T,$$

$$x(t) = \phi(t), \qquad -h \leq t \leq 0, \tag{6}$$

where $A_0(t)$, $A_1(t)$, and $B(t)$ are matrices of appropriate sizes whose elements represent the adjustable parameters of the plant. The initial function $\phi : [-h, 0] \to R^n$ is an element of $C([-h, 0]; R^n)$, the Banach space of continuous functions with norm $\|\phi\| = \max_{1\leq i\leq n} \sup_{t\in[-h,0]} |\phi_i(t)|$. Conditions of existence and uniqueness of solutions to delay-differential equations are covered in standard references of Hale [3], El'sgol'ts and Norkin [4]; they will be assumed throughout this work.

Let us first introduce a useful lemma which we need in establishing the controllability of a delay system. This lemma makes use of a sequence of matrices $\{Q_k(t)\}$ determined by the equations

$$Q_k(t) = A_0(t)Q_{k-1}(t) + A_1(t)Q_{k-1}(t-h), \qquad Q_0(t) = \begin{cases} B(t), & t > 0, \\ 0, & t \leq 0, \end{cases} \tag{7}$$

for $k = 1, 2, \ldots$, as introduced in [5,6].

LEMMA 1. *The linear system*

$$\dot{x}(t) = A_0(t)x(t) + A_1(t)x(t-h) + B(t)u(t) \tag{8}$$

is proper in R^n on the interval $(0, T]$ if and only if rank of $\{Q_0(t), Q_1(t), \ldots, Q_{n-1}(t)\}$ is n, for $t \in (0, T]$.

Proof. It is exactly as given by Manitius [7] for his Theorem 2.3.

Let the reference model to be followed be given as

$$\begin{aligned} \dot{x}_r(t) &= A_{0r}x_r(t) + A_{1r}x_r(t-h) + B_r u(t), \qquad 0 < t \leq T, \\ x_r(t) &= \phi_r(t), \qquad -h \leq t \leq 0, \end{aligned} \tag{9}$$

where A_{0r}, A_{1r} and B_r are constant matrices, and $\phi_r \in C([-h, 0], R^n)$. We assume that this reference system satisfies the conditions of Lemma 1, and thus it is controllable; and further we assume that it is stabilizable. Let us now denote the state error by $x_e(t) = x(t) - x_r(t)$. From (8) and (9) we obtain the state error equation as

$$\begin{aligned} \dot{x}_e(t) &= A_{0r}x_e(t) + A_{0e}(t)x(t) + A_{1r}x_e(t-h) + A_{1e}(t)x(t-h) + B_e(t)u(t), \\ x_e(t) &= \phi(t) - \phi_r(t), \qquad -h \leq t \leq 0, \end{aligned} \tag{10}$$

where $A_{0e}(t) = A_0(t) - A_{0r}$, $A_{1e}(t) = A_1(t) - A_{1r}$, and $B_e(t) = B(t) - B_r$. The plant follows the model if $x_e(t) \to 0$, $A_{0e}(t) \to 0$, $A_{1e}(t) \to 0$ and $B_e(t) \to 0$ as $t \to \infty$ which thus becomes the tracking objective. Let a functional $V(x_e)$ be given as

$$\begin{aligned} V(x_e) = {} & x_e^*(t)Px_e(t) + \int_{t-h}^{t} x_e^*(s)Rx_e(s)\,ds \\ & + \operatorname{tr}[A_{0e}(t)A_{0e}^*(t) + A_{1e}(t)A_{1e}^*(t) + B_e(t)B_e^*(t)], \end{aligned} \tag{11}$$

where P and R are some positive definite symmetric matrices and * denotes the transpose operation. Now if the matrices P and R are chosen to satisfy

$$\begin{pmatrix} A_{0r}^*P + PA_{0r} + R & PA_{1r} \\ A_{1r}^*P & -R \end{pmatrix} < 0, \tag{12}$$

then the adaptive laws can be formed as follows

(a) $\dot{A}_0(t) = -Px_e(t)x^*(t)$,

(b) $\dot{A}_1(t) = -Px_e(t)x^*(t-h)$, and

(c) $\dot{B}(t) = -Px_e(t)u^*(t)$.

With these conditions, the inequality $\dot{V}(x_e) < 0$ is satisfied which establishes the asymtotic stability. These conditions can be derived by observing that the time derivative of (11) along the trajectories of equation (10) can be written as

$$\begin{aligned} \dot{V}(t, x_e) = {} & x_e^*(t)[A_{0r}^*P + PA_{0r} + R]x_e(t) - x_e^*(t-h)Rx_e(t-h) \\ & + x_e^*(t)A_{1r}^*Px_e(t-h) + x_e^*(t-h)A_{1r}^*Px_e(t) \\ & + 2\operatorname{tr}[Px_e(t)x^*(t)A_{0e}^*(t) + \dot{A}_{0e}(t)A_{0e}^*(t)] \\ & + 2\operatorname{tr}[Px_e(t)x^*(t-h)A_{1e}^*(t) + \dot{A}_{1e}(t)A_{1e}^*(t)] \\ & + 2\operatorname{tr}[Px_e(t)u^*(t)B_e^*(t) + \dot{B}_e(t)B_e^*(t)]. \end{aligned} \tag{13}$$

Now it is easy to establish the stated conditions by writing the first four terms in a matrix form and replacing $\dot{A}_{ie} = \dot{A}_i$ for $i = 0, 1$.

III. TRACKING IN NONLINEAR SYSTEMS

Here we consider the plant described as

$$\dot{x}(t) = f(x(t), x(t-h), u(t)), \qquad 0 < t \leq T,$$

$$x(t) = \phi(t), \qquad -h \leq t \leq 0, \tag{14}$$

where $u(\cdot)$ is in appropriate control space and the initial function $\phi \in C([-h, 0], R^n)$ is as introduced in section II. We assume that system (14) is locally linearizable.

Let $\Omega = \{x : [-h, T] \to R^n\}$ be the state space of (14). Given a continuous function $u(\cdot)$ and an initial function $\phi \in C([-h, 0]; R^n)$, a function $x \in \Omega$ will be said to be the solution of (14) if $x(\cdot)$ is differentiable and satisfies (14) on $(0, T]$, and $x(t) = \phi(t)$ on $[-h, 0]$.

Let the $n \times n$ matrices $A_0(t)$ and $A_1(t)$, and $n \times m$ matrix $B(t)$ be defined as

$$A_{0ij}(t) = \frac{\partial f_i(\phi(0), 0, 0)}{\partial \phi_j(0)}, \quad A_{1ij}(t) = \frac{\partial f_i(0, \phi(-h), 0)}{\partial \phi_j(-h)}, \tag{15}$$

and

$$B_{ij}(t) = \frac{\partial f_i(0, 0, u(t))}{\partial u_j(t)}. \tag{16}$$

Note that these matrices are the parameters of a linear system that represents system (14) locally. Now let the reference model to be followed by the system (14) be given as

$$\dot{x}_{\mathrm{r}}(t) = f_{\mathrm{r}}(x_{\mathrm{r}}(t), x_{\mathrm{r}}(t-h), u(t)), \qquad 0 < t \leq T,$$

$$x_{\mathrm{r}}(t) = \phi_{\mathrm{r}}(t), \qquad -h \leq t \leq 0. \tag{17}$$

In the same neighborhood, let this reference model be represented by matrices $A_{0\mathrm{r}}$, $A_{1\mathrm{r}}$ and B_{r} derived similarly as in (15) and (16). We, therefore, can define the local error parameters $A_{0\mathrm{e}}(t) = A_0(t) - A_{0\mathrm{r}}$, $A_{1\mathrm{e}}(t) = A_1(t) - A_{1\mathrm{r}}$, and $B_{\mathrm{e}}(t) = B_{(}t) - B_{\mathrm{r}}$. We assume that the reference model is locally controllable, i.e. the conditions of Lemma 1 hold in a neighborhood of state space Ω. We also assume that the reference model is stabilizable, therefore there exists a Liapunov functional $V_{\mathrm{r}}(x_{\mathrm{r}})$ which is positive definite and has negative definite derivative such that

$$\dot{V}(x_{\mathrm{r}}) \leq -\alpha < 0 \tag{18}$$

along the trajectories of (17) in a small neighborhood of the origin in Ω.

Let x_r be the solution of (17) and define its ϵ-neighborhood $N_\epsilon(x_r) = \{x \in \Omega : ||x - x_r|| < \epsilon\}$, where $||\cdot||$ is norm in Ω. Let us introduce two sets in $M = \Omega - N_\epsilon(x_r)$ as

$$\mathcal{C} = \{x \in M : V_r(x) - V_r(x_r) > 0\}, \qquad \mathcal{D} = \{x \in M : V_r(x) - V_r(x_r) < 0\}. \tag{19}$$

Definition. We shall say that the system (14) is stabilized in Ω and tracks the reference model (17) if there exists an admissable control $u(t) = u_0(t)$, $t \geq 0$ and a time t_f such that $x \in N_\epsilon(x_r)$, for all $t \geq t_f$.

Consider two functionals $V(\cdot)$ and $W(\cdot)$ defined as follows:

$$V(x, x_r) = V_r(x) + V_r(x_r) + \tfrac{1}{2}\mathrm{tr}\{A_{0e}A_{0e}^* + A_{1e}A_{1e}^* + B_e B_e^*\}, \tag{20}$$

and

$$W(x, x_r) = |V_r(x) - V_r(x_r)| + \tfrac{1}{2}\mathrm{tr}\{A_{0e}A_{0e}^* + A_{1e}A_{1e}^* + B_e B_e^*\}, \tag{21}$$

for $x \in M$. Note that the functions $V(\cdot)$ and $W(\cdot)$ are positive definite. From the derivative of (20)

$$\dot{V}(x, x_r) = \dot{V}_r(x) + \dot{V}_r(x_r) + \mathrm{tr}\{\dot{A}_{0e}A_{0e}^* + \dot{A}_{1e}A_{1e}^* + \dot{B}_e B_e^*\}, \tag{22}$$

we see that if we form the adaptive law as

$$\dot{A}_{0e}A_{0e}^* + \dot{A}_{1e}A_{1e}^* + \dot{B}_e B_e^* = [\mathrm{grad} V_r(x_r)] f_r^*(x_r(t), x_r(t-h), u(t)), \tag{23}$$

we will have

$$\mathrm{tr}\{\dot{A}_{0e}A_{0e}^* + \dot{A}_{1e}A_{1e}^* + \dot{B}_e B_e^*\} = \dot{V}_r(x_r) \leq -\alpha. \tag{24}$$

With these conditions (22)-(24), a control law $u_0(t)$ can be designed to satisfy the criterion

$$[\mathrm{grad} V_r(x)]^* f(x(t), x(t-h), u_0(t)) \leq -\alpha. \tag{25}$$

Remark. Note that equation (25) gives a condition as specific as we can obtain without specifying $V_r(\cdot)$ and $f(\cdot)$.

Therefore the derivative of (21) satisfies

$$\dot{W}(x, x_r) = \begin{cases} \dot{V}_r(x)) = -\alpha, & x \in \mathcal{C}, \\ 2\dot{V}_r(x_r) - \dot{V}_r(x) \leq -\alpha, & x \in \mathcal{D}. \end{cases} \tag{26}$$

With this equation, all the sufficient conditions of Krasovskii are thus met.

IV. CONCLUSION

The known linear model reference adaptive control technique is extended to nonlinear functional differential systems. The method proposed uses state spaces of plant and reference model. Liapunov design technique is used to stabilize the system about a model.

Control scheme and adaptive laws are specified. The results apply to nonlinear functional differential systems with variable coefficients when the spectral analysis technique is in general not applicable. The method can be applied to various physical systems.

V. REFERENCES

[1] Landau,I.D., *Adaptive Control: the model reference approach*, M.Dekker, New York 1979

[2] Fiagbedzi,Y.A. and Pearson,A.E., "Finite dimensional approach to tracking in linear time lag systems", *Proceedings of the Control of Distributed Parameter Systems*, 4th IFAC Symposium, UCLA, Los Angeles 1986.

[3] Hale,J.K., *Functional Differential Equations*, Springer- Verlag , New York 1977.

[4] El'sgol'ts,L.E. and Norkin,S.B. *Introduction to the Theory and Application of Differential Equations with Deviating Arguments*, Academic Press, New York 1973.

[5] Zmood,R.B. "The Euclidean space controllability of control systems with delays", *SIAM J. Control, 12*, pp 609-623, 1974.

[6] Sinha,A.S.C., "Null-controllability of nonlinear infinite delay systems with restrained control", *Int. J. Control, Vol. 42*, pp 735-741, 1985.

[7] Manitius,A., *Control Theory and Topics in Functional Analysis, Vol. III*, International Atomic Energy Agency, pp 43-178, 1976.

Fuzzy Gain Scheduling Control of A Non-linear HVAC System

Guang Geng and A. L. Dexter

Department of Engineering Science, University of Oxford

Parks Road, Oxford OX1 3PJ, U. K.

Abstract

The paper examines the use of gain scheduling methods to deal with processes that have input non-linearities. A fuzzy gain scheduling scheme has been developed and cascaded with two different types of controllers; a conventional proportional plus integral and a self-tuning predictive controller. A detailed simulation of an air-conditioning system has been used to test the scheme. Results show that when the fuzzy gain scheduling scheme is used in an inner-loop, it improves the overall performance of the system, as well as that of the inner-loop.

Nomenclature

T_{ai}	Temperature of inlet air;
T_{as}	Temperature of supply air;
T_{asd}	Desired temperature of supply air;
T_{az}	Temperature of zone air;
T_{azd}	Desired temperature of zone air;
T_{hws}	Temperature of hot water supplied by the boiler;
T_{hwr}	Temperature of returned hot water;
T_{cws}	Temperature of cold water supplied by the chiller;
T_{cwr}	Temperature of returned cold water;
U	Stem position of the control valves.

1 Introduction

The paper examines the use of gain scheduling methods to deal with processes that have input non-linearities. A fuzzy gain scheduling scheme has been developed and cascaded with two different types of controllers; a conventional proportional plus integral (PI) and a self-tuning predictive controller. The work described in the paper is part of a project to develop self-adaptive controllers to operate the Heating, Ventilating and Air-Conditioning (HVAC) systems in commercial buildings.

The control system of the air-handling plant for a large building must be designed to cope with a wide range of operating conditions, since the internal heat loads and the external environment can change significantly from day to day and from season to season. The uncontrolled response of the zone air temperature to typical disturbances is shown in Figure 1. Severe non-linearities and significant disturbances can occur in HVAC plants [1]. A cascaded control system is often used to reduce the effects of the non-linearities by regulating the supply air temperature, as shown in Figure 2.

The aim of this work is to design an inner-loop controller capable of regulating the temperature of the supply air T_{as} leaving the air-handler by adjusting the stem position U of the control valves of the cooling and heating coils.

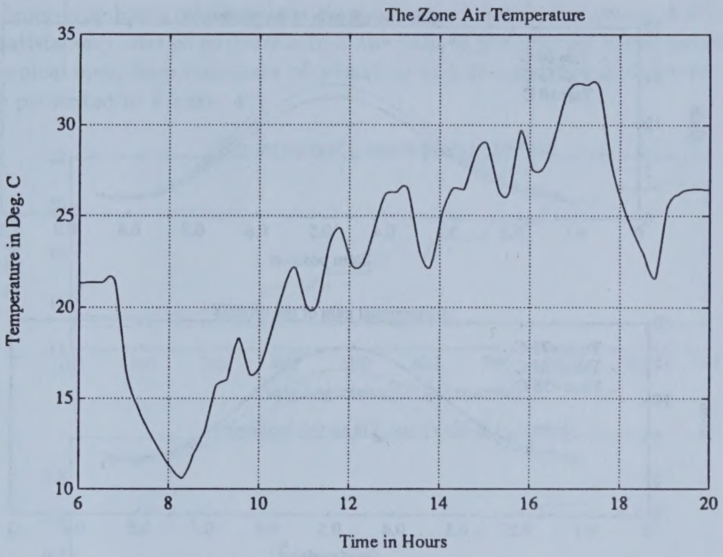


Figure 1: Typical uncontrolled response of the air temperature in the zone

In an air-handling unit, the non-linearities exist mainly in the relationship between T_{as}, the stem position of the control valve U and the temperatures T_{hws}, T_{cws} and T_{ai}. The variations in the incremental steady-state gain of the simulated heating coil with the stem position, the supply water temperature and the inlet air temperature are shown in Figure 3

The effects of the changes in the temperature of hot water supplied by the boiler on the incremental steady-state gain are significant only when the valve is half open; on the other hand, the effect of changes in the temperature of the inlet air are relatively small even when the valve is half open.

Experience on an experimental air-handling unit at Oxford has shown that even well

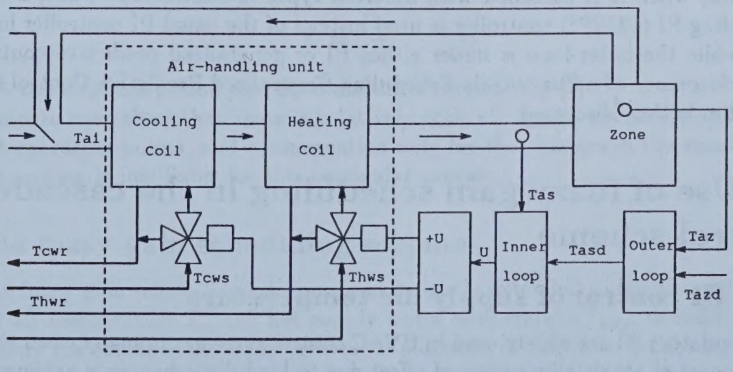

Air-handling unit
Cooling
Coil
Heating
Coil
Zone
Tai
Tas
Tcwr
Tcws
Thws
Thwr
+U
-U
U
Inner
loop
Tasd
Outer
loop
Taz
Tazd

Figure 2: Block diagram of the air-conditioning system

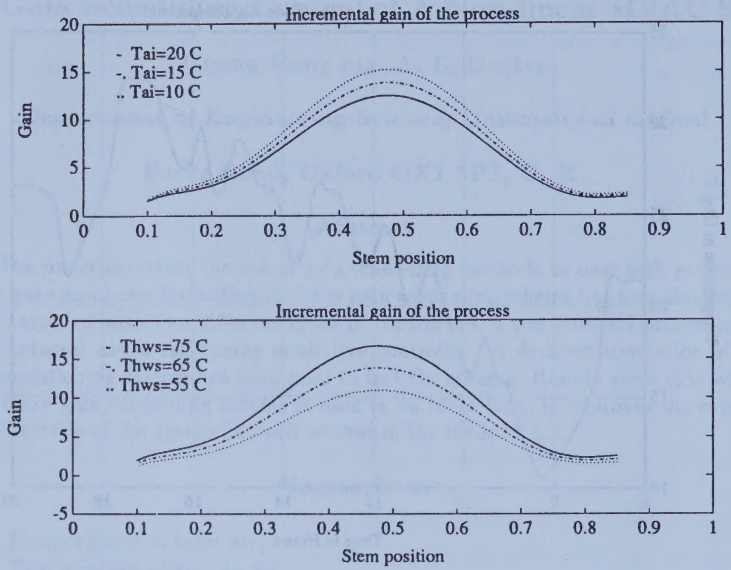


Figure 3: A typical variation in steady-state gains of the heating coil

designed model-based adaptive controllers can sometimes fail to control properly. Self-tuners based on linear models have shown their ability to achieve adequate control for special classes of process, such as those which are slowly time-varying and weakly non-linear. However, for HVAC processes that are highly non-linear, the performance of these self-tuners can be disappointing [2]. Gain scheduling makes it possible to compensate for the changes of gain at different operating points [3], and the adoption of the fuzzy approach ensures that the compensation is smooth and reasonable [4]. In addition, the use of if-then structured rules makes it easier to incorporate expert knowledge about the process [5].

A series of experiments have been performanced to examine the behaviour of such a scheme, when it is cascaded with different types of controllers. First, a Fuzzy Gain Scheduling PI (FGSPI) controller is used instead of the usual PI controller in the inner-loop, while the outer-loop is under either PI or generalized predictive control (GPC); the performance of a Fuzzy Gain Scheduling Generalized Predictive Control (FGSGPC) algorithm is then discussed.

2 Use of fuzzy gain scheduling in the cascaded control scheme

2.1 PI control of supply air temperature

PID regulators [6] are widely used in HVAC control systems. In many cases the principal requirement of stable elimination of offset due to load-disturbances is achieved by PI action alone, though derivative action is occasionally added for plants with excessive phase lag. In this case, a PI controller was chosen for the supply air temperature control loop, since the inner-loop has a relative short time delay and simple dynamics. A PI controller can give satisfactory overall performance if the gain of the process is compensated sufficiently. Typical open-loop responses of a heating coil at a number of different operating points are presented in Figure 4

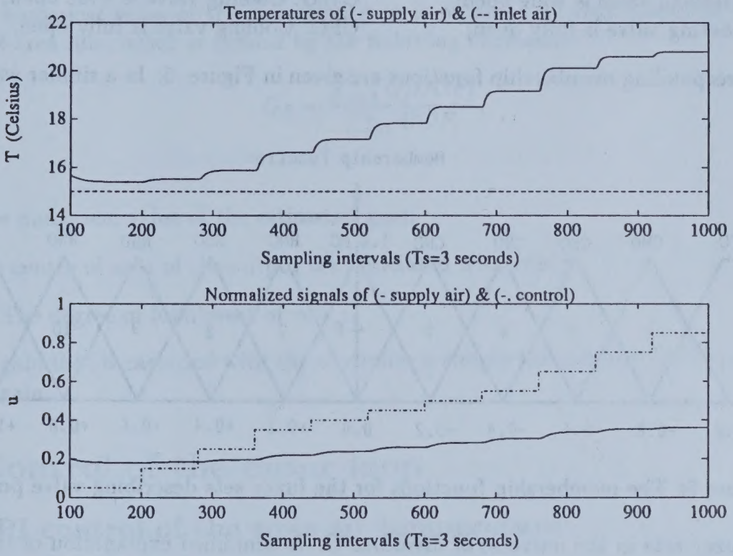


Figure 4: The open-loop response of the heating coil at different operating points

The PI controller is tuned using Ziegler-Nichols' rules based on estimates of the steady-state gain, the time delay, the dominant time constant. If the PI controller is of the form

$$u(s) = K(1 + \frac{1}{T_i s}), \tag{1}$$

its parameter are calculated by

$$K = 0.45 S_u, \tag{2}$$

$$T_i = \frac{P_u}{1.2}, \tag{3}$$

where P_u is the ultimate period and S_u the ultimate sensitivity [7].

Experiments have show that there is relatively little change in the process dynamics at different operating points, and compensation only for the changes in the steady-state gain of the process is justifiable for this particular process.

2.2 The fuzzy gain scheduling scheme

Though the fuzzy gain scheduling scheme could be easily extended to incorporate effects of the inlet air temperature T_{ai} and hot supply water temperature T_{hws}, to illustrate the approach, only the effects of different valve stem positions (a universe of discourse U) on the steady-state gain (a universe of discourse G) are considered here. In the universe of discourse U, the following fuzzy sets are defined,

BC: Both valves are closed;
HNC: Heating valve is nearly closed;
HSO: Heating valve is slightly open;
HHO: Heating valve is half open;
HWO: Heating valve is wide open;
HFO: Heating valve is fully open;
CNC: Cooling valve is nearly closed;
CSO: Cooling valve is slightly open;
CHO: Cooling valve is half open;
CWO: Cooling valve is wide open;
CFO: Cooling valve is fully open.

The corresponding membership functions are given in Figure 5. In a similar way, we can

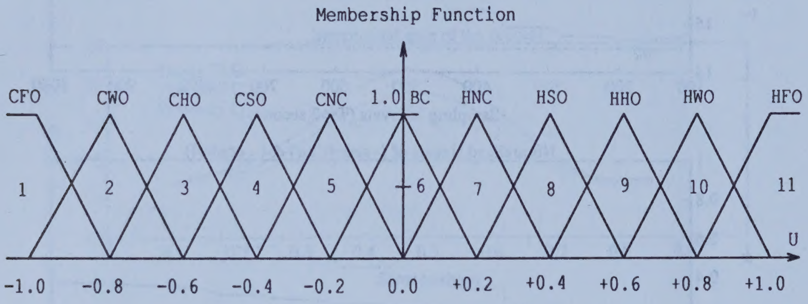


Figure 5: The membership functions for the fuzzy sets describing valve position

define fuzzy sets in the universe of discourse G. A simplified explanation of the rules is given here using only four sets to describe the gain and assuming that the heating and cooling coils have similar characteristics.

The fuzzy gain scheduling rules are constructed as follows:

Rule 1: **IF** BFC **THEN** process gain is very low;
Rule 2: **IF** HNC **OR** CNC **THEN** process gain is low;
Rule 3: **IF** HSO **OR** CSO **THEN** process gain is medium;
Rule 3: **IF** HHO **OR** CHO **THEN** process gain is high;
Rule 4: **IF** HWO **OR** CWO **THEN** process gain is medium;
Rule 5: **IF** HFO **OR** CFO **THEN** process gain is very low;

which are of the form,

IF A **THEN** D,

where A.is the condition set and D is the output set. Notice if the gain characteristics of the heating and cooling coils are different, more rules are needed to reflect these differences; and if the other factors are included in the gain scheduling scheme, the rules must be expressed in the form

IF A **AND** B **THEN** D.

Thus, a particular value of the stem position of the control valve u will generate values for the membership functions $\mu_{BC}(u), \mu_{HNC}(u), \mu_{HSO}(u)$, etc., which are indications of to what extent u belongs to each of the fuzzy sets in the universe of discourse U.

For the simple single-input single-output fuzzy gain scheduling algorithm, the Degree Of Fulfillment (DOF) of the scheduling rule j is determined by

$$DOF_j = \mu_i(u),$$

where $\mu_i(u)$ is the membership value of the input fuzzy set which forms the condition part of the rule j. For the membership functions used here, there are at most two rules whose DOF will be non-zero for a given value of u.

The numerical value of the estimated gain is obtained by defuzzification using the center-of-area rule, which is defined by the following expression:

$$G_P = \frac{\sum_{i=1}^{5} G_j DOF_j}{\sum_{i=1}^{5} DOF_j}, \tag{4}$$

where

G_P: The numerical value of the estimated gain;

G_j: The centre of area of the output set associated with rule j;

DOF_j: The degree of fulfillment of rule j.

The gain that is cascaded with the controller is simply the reciprocal of the estimated process gain.

3 Control of the outer loop

3.1 PI control of the zone air temperature

The dynamics of the outer-loop are very different from the dynamic characteristics of the inner-loop and have a significant element of time delay. Therefore direct application of the Ziegler-Nichols' rule can lead to an over-sluggish response. For substantial time delays of the order of, or larger than, the slowest process mode, the integral time can be determined [8] according to the equation

$$T_i \simeq 0.5 P_u. \tag{5}$$

Automation of tuning the rules for the controller is at present under investigation.

3.2 Predictive control of the zone air temperature

The application of predictive controllers may be feasible if a reasonably reliable process model can be estimated. A GPC controller has been used at the outer-loop level. Details about the settings of the controller are given in Dexter and Haves [2] and are not repeated here in this paper.

4 Discussion of the results

A series of simulations have been carried out, using the simulation package HVACSIM+ [9] developed at the National Bureau of Standards to allow detailed simulation of the thermal behaviour of the building, plant and controllers.

Figure 6 shows the behaviour of the inner-loop controllers as they follow the changes of set-point which are generated by the outer-loop controller. Clearly, FGSPI control gives better performance.

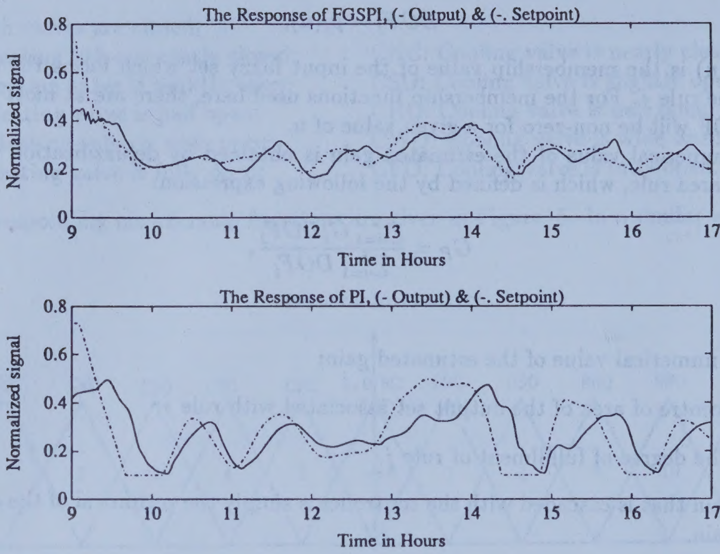


Figure 6: Behaviour of the inner-loop with and without fuzzy gain scheduling

Figure 7 shows that better control of the supply air temperature does result in improved control of the zone air temperature and that the overall performance is further improved by the use of GPC in the outer-loop. Although the results have demonstrated the benefits of generalized predictive control, more recent work has suggested that a PI controller cascaded with a Smith predictor may perform equally well.

5 Direct integration of fuzzy gain scheduling scheme with the GPC algorithm

The feasibility of integrating fuzzy gain scheduling scheme with a generalized predictive controller is also being examined. Again, the controller is similar to that described by Dexter and Haves [2]. There are two ways in which the fuzzy gain scheduling scheme might be incorporated. First, as in case of PI control, it is used to compensate the gain of the process by cascading it with the GPC algorithm. The problem of this approach is that the potential advantage of using GPC may be lost, since any serious mismatch in the process model can lead to very poor performance. Preliminary results have shown that oscillatory control signals can be generated, if care is not taken in selecting the tuning knobs of the controller. Alternatively, the fuzzy gain scheduling approach may be extended to use different models at different operating points. For example, using the simple first order model

$$(1 + a_1 q^{-1})y(t) = b_0 u(t - k_d)$$

the parameter b_0 might be determined according to the operating point. One of the advantages of using the latter method is that knowledge about the variations in the other parameter of the model a_1 can also be included in the controller. Preliminary

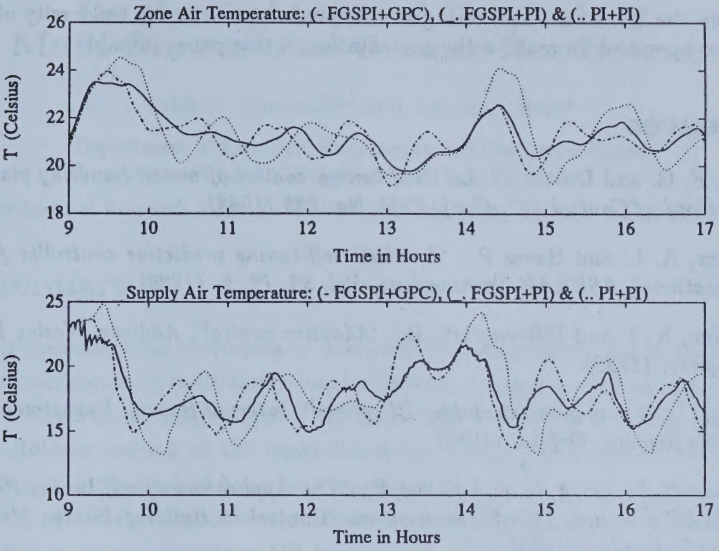


Figure 7: Cascaded control of zone air temperature

results have demonstrated the potential of the latter method. A further comparison of these methods is now under investigation.

6 Conclusions

The overall performance of the control system, as well as that of the inner-loop is improved, when fuzzy gain scheduling scheme is cascaded with the controller in the inner-loop. When self-tuning controllers are used in the outer-loop, the scheduling scheme improves the environment for the parameter estimator, which could lead to better control in the outer-loop. The procedures for automatic determination of process gain and the selection of fuzzy variables and their membership functions are not addressed in this paper, but are the subjects of current research.

The work has concentrated on finding a practical and economical solution to problem of controlling an HVAC system. A modified Smith predictor is to be used to improve the performance of a PI controller in the outer-loop, and a comparison will be made with that of GPC, when the same model is used in both controllers.

7 Acknowledgements

Mr. G. Geng was supported by SBFSS and ORS scholarship schemes. The presentation of this paper is supported by Linacre College, the Department of Engineering Science, the Graduate Studies Committee of Oxford and the British Council. The authors would like to acknowledge Dr. P. Haves for the time and effort that he has spent in supporting the simulation work. Mr. Geng would also like to thank Dr. J. Bumby and Mr. G. M.

Geary from the School of Engineering and Applied Science in the University of Durham for their co-operation in making the presentation of this paper possible.

References

[1] Jota, F. G. and Dexter, A. L.: *'Self-tuning control of an air-handling plant', Proc. IEE Conf., "Control 88", Conf. Publ. No. 285 (1988).*

[2] Dexter, A. L. and Haves P.: *'A robust self-tuning predictive controller for HVAC applications', ASHRAE Transactions, Vol. 95, Pt. 2, (1989).*

[3] Åström, K. J. and Wittenmark, B.: *'Adaptive control', Addison-Wesley Publishing Company, (1989).*

[4] Geng, G.: *'Fuzzy gain scheduling PI control', Internal Report, Department of Engineering Science, Oxford, (1989).*

[5] Dexter, A. L., Geng, G. and Haves, P.: *'The Application of self-tuning PID control to HVAC systems', IEE Colloquium on "Control in Building Energy Management System", London (1990).*

[6] Clarke, D. W.: *'PID algorithms and their computer implementation', Trans. Inst. Meas. Contr.,6(6) (1985).*

[7] Ziegler, J. G., Nichols, N. B. and Rochester, N. Y.: *'Optimum settings for automatic controllers', Translation ASME, 64, pp.759-765, (1942).*

[8] Vit, K.: *'Design of delayed control systems', Proc. IEE, Vol. 134, Pt. D. No. 2, pp124-128, (1987).*

[9] Park, C., Clarke D. R. and Kelly, G. E.: *'An overview of HVACSIM+: A dynamic building/HVAC/control system program', Proc. 1st Annual Building Energy Simulation Conf., Seattle, U. S. A. (1985).*

On the Tightness of Constrained Multi-Dimensional Cramer-Rao Bounds

John D. Gorman[1,2,3] and Alfred O. Hero[1]

[1]Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109,

[2]Environmental Research Institute of Michigan, Box 8618, Ann Arbor, MI 48107-8618

1 Summary

This paper presents some extensions of previously obtained results [6] on the impact of parameter constraints on multidimensional parameter estimator performance.

In [6] a constrained Cramér-Rao (CR) bound was derived as a certain limit of the Chapman-Robbins version of the multi-dimensional Barankin bound. This limit was taken with respect to a set of linearly independent "test points" converging to the true parameter $\underline{\theta}$ within the constrained parameter space. The constrained CR bound was implemented as an unconstrained CR bound with a rank-reduced Fisher Information matrix formed by projecting the rows and columns of the unconstrained Fisher information matrix onto the plane tangent to the constrained parameter space. Based on these results it was concluded that: (i) inequality-type constraints which do not reduce the dimension of the unconstrained parameter space in the neighborhood of $\underline{\theta}$ do not change the CR bound relative to the unconstrained case; (ii) equality-type constraints which reduce the dimension of the unconstrained parameter space in the neighborhood of $\underline{\theta}$ achieve a reduction in the CR bound; (iii) the constrained CR bound is achievable for linear constraints and exponential families of densities.

In this paper we characterize the tightness of the constrained CR bound derived in [6]. First we define a best *local* Barankin bound of the Chapman-Robbins type. Then we examine conditions under which the constrained CR bound is the best local Barankin bound. In particular, we note that the constrained CR bound is only guaranteed to be the tightest local Barankin bound if either: the constraints are linear and the density lies in the exponential family; or if the estimator is unbiased. We then provide a simple counter-example showing lack of tightness when the density is of the exponential family but the constraint is non-linear in $\underline{\theta}$. Motivated by these results we examine approaches for obtaining sharper constrained bounds which incorporate higher order information and account for the curvature of the constraint set in addition to its tangent plane. Our approach can be related and contrasted to methods [2, 4, 8] for studying the asymptotic (large sample) distribution of constrained parameter estimators in curved exponential families.

---

[3] Supported under the ERIM/University of Michigan Doctoral Fellowship Program.

2 Previous Results

$\underline{\theta}$ and $[\theta_i]_{i=1,\dots,n}$ will denote a column vector, $[\theta_1,\dots,\theta_n]^T$, of unknown but nonrandom parameters contained in the unconstrained parameter space $\Theta = \mathbb{R}^n$. $(\Omega, \mathcal{F}, \mu)$ is a measure space, Θ is a nonempty n-dimensional subset of $\mathbb{R}^n$, and, for each $\underline{\theta} \in \Theta$, $P_{\underline{\theta}}$ is a probability distribution on $\mathcal{F}$ parameterized by $\underline{\theta}$. Furthermore, it is assumed that each $P_{\underline{\theta}}$ possesses a density $f_{\underline{\theta}}$ with respect to reference measure μ. $\mathrm{E}_{\underline{\theta}}$ denotes expectation with respect to $P_{\underline{\theta}}$. $\underline{X}$ is a random vector taking values $\underline{x} \in \Omega$ according to the distribution $P_{\underline{\theta}}$. Let $\underline{\tau}_{\underline{\theta}}$ be a differentiable vector function of $\underline{\theta}$ and let $\underline{t}(\underline{X})$ denote an estimator for $\underline{\tau}_{\underline{\theta}}$ based upon the observation $\underline{X}$. The vector mean of this estimator will be denoted $\underline{m}_{\underline{\theta}} \stackrel{\mathrm{def}}{=} \mathrm{E}_{\underline{\theta}}\{\underline{t}(\underline{X})\}$. The parameter space of interest, defined by functional constraints or side information, is $\Theta_C \subset \Theta$. We assume that Θ_C is defined as $\Theta_C = \{\underline{\theta} \in \mathbb{R}^n : \mathcal{G}_{\underline{\theta}} = 0\}$, for continuously-differentiable $\mathcal{G} : \mathbb{R}^n \to \mathbb{R}^k$. Our objective will be to investigate the impact of parameter constraints on bounds for the minimum mean-squared estimation error, where error is measured by the covariance matrix: $\Sigma_{\underline{\theta}} \stackrel{\mathrm{def}}{=} \mathrm{E}_{\underline{\theta}}\{(\underline{t}(\underline{X}) - \underline{m}_{\underline{\theta}})(\underline{t}(\underline{X}) - \underline{m}_{\underline{\theta}})^T\}$. We say that a matrix B is a lower bound on a matrix A if $A \geq B$ in the sense that $A - B$ is non-negative definite.

Let $\underline{\theta}, \underline{\theta}_1, \dots, \underline{\theta}_k \in \Theta_C$ be such that each point $\underline{\theta}_i$ has the representation $\underline{\theta}_i = \underline{\theta} + \Delta_i \underline{\nu}_i$, where $\underline{\nu}_i$ is a unit vector called a *direction vector* and $\Delta_i > 0$. The points $\underline{\theta}_i \in \Theta_C$ are referred to as *test points.* Define the k-length row vector $\delta f_{\underline{\theta}} \stackrel{\mathrm{def}}{=} [\delta_1 f_{\underline{\theta}}, \dots, \delta_k f_{\underline{\theta}}]$ and the $n \times k$ matrix $\delta \underline{m}_{\underline{\theta}} \stackrel{\mathrm{def}}{=} [\delta_1 \underline{m}_{\underline{\theta}}, \dots, \delta_k \underline{m}_{\underline{\theta}}]$, where respectively, $\delta_i f_{\underline{\theta}} \stackrel{\mathrm{def}}{=} \Delta_i^{-1}(f_{\underline{\theta}_i} - f_{\underline{\theta}})$ and $\delta_i \underline{m}_{\underline{\theta}} \stackrel{\mathrm{def}}{=} \Delta_i^{-1}(\underline{m}_{\underline{\theta}_i} - \underline{m}_{\underline{\theta}})$. Define the matrix

$$J_{\underline{\theta}}(\Delta) = \mathrm{E}_{\underline{\theta}} \left[\frac{\delta f_{\underline{\theta}}}{f_{\underline{\theta}}} \right]^T \left[\frac{\delta f_{\underline{\theta}}}{f_{\underline{\theta}}} \right]. \tag{1}$$

Then we have the following lower bound on estimation error covariance:

Proposition 1 *[6]:*

$$\Sigma_{\underline{\theta}} \geq B(\underline{\theta}, \underline{\theta}_1, \dots, \underline{\theta}_k), \tag{2}$$

where:

$$B(\underline{\theta}, \underline{\theta}_1, \dots, \underline{\theta}_k) = [\delta \underline{m}_{\underline{\theta}}] \cdot J^{+}_{\underline{\theta}}(\Delta) \cdot [\delta \underline{m}_{\underline{\theta}}]^T, \tag{3}$$

and "+" denotes (the Moore-Penrose) pseudo-inverse.

This is a particular version of the more general Barankin lower bound [3] on error covariance, and like the Barankin bound, is a global bound in the sense that the greatest lower bound of this type [cf. equation (3)] is a supremum taken over the class of all finite sets of points chosen from Θ_C. We define a *local* version of this bound by evaluating the bound over a local neighborhood of the point $\underline{\theta}$ and allowing the diameter of this neighborhood to shrink to zero within Θ_C:

$$\Sigma_{\underline{\theta}} \geq \text{limsup}_{\Delta_1,\ldots,\Delta_k \to 0} B(\underline{\theta}, \underline{\theta} + \Delta_1 \underline{\nu}_1, \ldots, \underline{\theta} + \Delta_k \underline{\nu}_k), \tag{4}$$

where the limsup is taken over all finite sets of test points converging to $\underline{\theta}$ along continuously differentiable paths. We call the greatest lower bound of this type the *best local* Barankin bound.

In [6], we showed that for the appropriate choice of test points, the quadratic form in (3) converges to what we call the *constrained* CR bound. The constrained CR bound is of the same form as the standard CR bound, except that the Fisher information matrix is replaced by a rank-reduced Fisher information matrix formed by projecting the rows and columns of the unconstrained Fisher information matrix onto the tangent hyperplane to the constraint set Θ_C. An equivalent representation of the constrained CR bound is:

$$\Sigma_{\underline{\theta}} \geq \nabla \underline{m}_{\underline{\theta}} Q_{\underline{\theta}} J_{\underline{\theta}}^{-1} \nabla \underline{m}_{\underline{\theta}}, \tag{5}$$

where $\nabla \underline{m}_{\underline{\theta}}$ denotes the gradient of the mean vector and $J_{\underline{\theta}}$ is the Fisher information matrix. The matrix $Q_{\underline{\theta}}$ is an oblique projection operator whose range is the tangent hyperplane to the constraint set at the point $\underline{\theta}$; see [6] for details. For equality constraints, where the dimension of the parameter space is reduced, the constrained bound is smaller than the unconstrained bound, indicating a possible reduction in the variance. For inequality constraints, the dimension of the parameter space is unchanged and the constrained and unconstrained bounds are equivalent. More specifically, in [6] we show that when $\underline{\tau}_{\underline{\theta}} = \underline{\theta}$ and the density $f_{\underline{\theta}}$ lies in an exponential family, the constrained CR bound is the best local Barankin bound if either:

(i) the constraints $G_{\underline{\theta}}$ are linear, or

(ii) $\underline{t}(\underline{X})$ is an unbiased estimate of $\underline{\theta}$.

Here we show by simple counterexample that it is possible for the bound in (4) to converge to a limit which is strictly greater than the constrained CR bound. The result is that the constrained CR bound is not, in general, the tightest bound of the form (4). Unlike standard treatments [3, 5, 7], we formulate the bound using the pseudo-inverse of $J_{\underline{\theta}}(\Delta)$ rather than its inverse since $J_{\underline{\theta}}(\Delta)$ can become singular in the limit as Δ goes to zero. It is straightforward to show that for densities $f_{\underline{\theta}}$ belonging to an exponential family and linearly-independent direction vectors $\{\underline{\nu}_1, \ldots, \underline{\nu}_k\}$, and $\Delta_i > 0$, the matrix $J_{\underline{\theta}}(\Delta)$ is strictly positive-definite. The constraint $\underline{\theta}_i = \underline{\theta} + \Delta_i \underline{\nu}_i \in \Theta_C$, however, implies that in the limit there are at most k linearly independent direction vectors $\underline{\nu}_i$. Thus in the limit $\Delta \downarrow 0$, $J_{\underline{\theta}}(\Delta)$ converges to the projection of the rows and columns of the Fisher information matrix onto the tangent hyperplane Θ_C at the point $\underline{\theta}$ [6]. This projection is rank deficient, and hence, because of the discontinuity associated with the pseudo-inverse at zero [9], $J_{\underline{\theta}}^{+}(\Delta)$ becomes unbounded as the $\Delta \downarrow 0$. Eigenvalues of

$J^{+}_{\underline{\theta}}(\Delta)$ associated with eigenvectors lying in the tangent plane are well behaved, but the remaining eigenvalues are unbounded. As a result, it is possible for the limit of the quadratic form (3) to converge to a bound *strictly greater* than the constrained CR bound. This behaviour is illustrated in the following example.

3 A simple counterexample: Normal Circle

We consider the case of estimating the length the mean vector of a bivariate Gaussian random variable subject to the constraint that the mean vector lies on a circle. Let $\underline{\theta} = [\theta_1, \theta_2]$ and let $\underline{X} \in \mathbb{R}^2$ be a Gaussian random vector with mean $\underline{\theta}$ and covariance $K = \mathrm{diag}\{\sigma_1^2, \sigma_2^2\}$. Let $\underline{\tau_\theta} = \underline{\theta}^T \underline{\theta} = \|\underline{\theta}\|^2$, the squared Euclidean length of $\underline{\theta}$, and consider lower bounding the variance of estimators $\underline{t}(\underline{X})$ of $\underline{\tau_\theta}$ when $\underline{\theta}$ is known to lie on the circle

$$\Theta_C = \{\underline{\theta} : (\underline{\theta} - \underline{c})^T (\underline{\theta} - \underline{c}) = r^2\}, \quad \underline{c} = [c_1, c_2]. \tag{6}$$

For $\underline{\theta} \in \Theta_C$, X has density

$$f_{\underline{\theta}}(\underline{x}) = (2\pi)^{-1} |K|^{-\frac{1}{2}} \exp\left\{\frac{1}{2}(\underline{\theta} - \underline{x}) K^{-1} (\underline{\theta} - \underline{x})\right\}. \tag{7}$$

The constrained CR bound for this example is:

$$\Sigma_{\underline{\theta}} \geq 4\theta_1^2 \tag{8}$$

Let $\underline{\theta}, \underline{\theta}_i \in \Theta_C, i = 1, \ldots, k$ and assume $\Delta_i = \|\underline{\theta}_i - \underline{\theta}\|$ for each i. Note that if we define the function

$$G(\underline{\theta}_i, \underline{\theta}_j; \underline{\theta}) = \int_\Omega \frac{f_{\underline{\theta}_i}(\underline{x}) f_{\underline{\theta}_j}(\underline{x})}{f_{\underline{\theta}}(\underline{x})} d\mu(\underline{x}). \tag{9}$$

Then the matrix $J_{\underline{\theta}}(\Delta)$ can be rewritten as:

$$J_{\underline{\theta}}(\Delta) = \left[\frac{1}{\Delta_i \Delta_j}\{G(\underline{\theta}_i, \underline{\theta}_j; \underline{\theta}) - 1\}\right]_{i,j=1,\ldots,k}. \tag{10}$$

This relationship is sometimes useful in computing the Barankin bound. For this example G is given by:

$$\begin{aligned}
G(\underline{\theta}_i, \underline{\theta}_j; \underline{\theta}) &= \int_\Omega \frac{f_{\underline{\theta}_i}(\underline{x}) f_{\underline{\theta}_j}(\underline{x})}{f_{\underline{\theta}}(\underline{x})} d\mu(\underline{x}) \\
&= \int_\Omega (2\pi)^{-1} |K|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\underline{x} - \underline{\theta}_i)^T K^{-1} (\underline{x} - \underline{\theta}_i)\right. \\
&\quad \left. -\frac{1}{2}(\underline{x} - \underline{\theta}_j)^T K^{-1} (\underline{x} - \underline{\theta}_j) + \frac{1}{2}(\underline{x} - \underline{\theta})^T K^{-1} (\underline{x} - \underline{\theta})\right\} d\mu(\underline{x}) \\
&= \exp\{\Delta_i \Delta_j \underline{\nu}_i K^{-1} \underline{\nu}_j\}
\end{aligned}$$

where $\underline{\xi} = \underline{\theta}_i + \underline{\theta}_j - \underline{\theta}$. The matrix $J_{\underline{\theta}}(\Delta)$ can then be written as:

$$\begin{aligned} J_{\underline{\theta}}(\Delta) &= \frac{1}{\Delta_i \Delta_j}\left[\exp\{\Delta_i \Delta_j \underline{\nu}_i K^{-1} \underline{\nu}_j\} - 1\right] \\ &= \underline{\nu}_i K^{-1} \underline{\nu}_j + \underline{\underline{O}}(\Delta^2), \end{aligned} \tag{11}$$

where $\underline{\underline{O}}(\Delta^2)$ is a matrix with terms of order $O(\Delta^2)$.

Consider evaluating the limit in (4) over all continuously differentiable paths through the point $\underline{\theta} = [c_1, c_2 - r]$. Assume r is fixed and $\infty > r > \Delta$. We choose points $\underline{\theta}_i \in \Theta_C$ that converge to $\underline{\theta}$ along a quadratic path as $\Delta \downarrow 0$. This is in contrast to [6] where convergence over line segments was studied. First we show that a sequence of Barankin-type bounds based on one test point converges to the constrained CR bound. Next, we show that a sequence of bounds based upon two test points does not converge to the constrained CR bound, but instead, to a limit strictly greater than the constrained CR bound; this is because of the discontinuity of the pseudo-inverse as eigenvalues of the matrix $J_{\underline{\theta}}(\Delta)$ tend to zero.

A single test point

Suppose $\underline{\theta} = [c_1, c_2 - r]$, and $\underline{\theta}_1 = \underline{\theta} + \Delta \underline{\nu}_1(\Delta)$, where

$$\underline{\nu}_1(\Delta) = \frac{\sqrt{r^2 - \Delta^2}}{r}\underline{e}_1 + \frac{\Delta}{r}\underline{e}_2, \tag{12}$$

and $\underline{e}_1$ and $\underline{e}_2$ are the standard basis vectors. Substitution of (12) in (11) along with straightforward calculations yields:

$$J_{\underline{\theta}}(\Delta) = \sigma_1^{-2}\left(\frac{r^2 - \Delta^2}{r^2}\right) + \sigma_2^{-2}\left(\frac{\Delta^2}{r^2}\right)\left(\frac{\sigma_1^{-4} r^2}{2} + 1\right) + \underline{\underline{O}}(\Delta^4). \tag{13}$$

Similarly for an unbiased estimator of $\tau_{\underline{\theta}}$:

$$\delta \underline{m}_{\underline{\theta}} = 2\theta_1 \frac{\sqrt{r^2 - \Delta^2}}{r} + \Delta. \tag{14}$$

The limit of (3) exists and is given by the constrained CR bound [6] as indicated in (8):

$$\Sigma_{\underline{\theta}} \;\geq\; \lim_{\Delta \to 0}[\delta \underline{m}_{\underline{\theta}}] \cdot J_{\underline{\theta}}^{+}(\Delta) \cdot [\delta \underline{m}_{\underline{\theta}}]^T = 4\theta_1 \sigma_1^2. \tag{15}$$

Two test points

Suppose, however, that *two* test points are chosen, $\underline{\theta} + \Delta \underline{\nu}_1(\Delta)$ and $\underline{\theta} + \Delta \underline{\nu}_2(\Delta)$, where:

$$\underline{\nu}_2(\Delta) = -\frac{\sqrt{r^2 - \Delta^2}}{r}\underline{e}_1 + \frac{\Delta}{r}\underline{e}_2. \tag{16}$$

Then the term $J_{\underline{\theta}}(\Delta)$ in the Chapman-Robbins bound (10) can be computed as:

$$\begin{aligned} J_{\underline{\theta}}(\Delta) &= \sigma_1^{-2}\left(\frac{r^2-\Delta_1^2}{r^2}\right)\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \\ &\quad + \sigma_2^{-2}\left(\frac{\Delta_1^2}{r^2}\right)\left(\frac{\sigma_1^{-4}r^2}{2}+1\right)\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \underline{\underline{Q}}(\Delta^4), \end{aligned} \tag{17}$$

where $\underline{\underline{Q}}(\Delta^4)$ is a matrix with elements of order $O(\Delta^4)$. Its inverse is:

$$\begin{aligned} J_{\underline{\theta}}^{+}(\Delta) &= \frac{\sigma_1^2}{4}\left(\frac{r^2}{r^2-\Delta_1^2}\right)\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \\ &\quad + \frac{\sigma_2^2}{4}\left(\frac{r^2}{\Delta_1^2}\right)\left(\frac{2\sigma_1^4\sigma_2^2}{2\sigma_1^4+\sigma_2^2r^2}\right)\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \underline{\underline{Q}}(\Delta^4). \end{aligned} \tag{18}$$

Note that the coefficient of the second term in (18) above is unbounded as Δ goes to zero. Also, for an unbiased estimator,

$$\delta \underline{m}_{\underline{\theta}} = 2\theta_1\frac{\sqrt{r^2-\Delta^2}}{r}\begin{pmatrix} 1 & -1 \end{pmatrix} + (2\theta_2+r)\frac{\Delta}{r}\begin{pmatrix} 1 & 1 \end{pmatrix}. \tag{19}$$

In the limit, the covariance bound (3) is strictly greater than the constrained CR bound (8):

$$\Sigma_{\underline{\theta}} \geq 4\theta_1^2\sigma_1^2 + 2\left(2\theta_2+r\right)^2\left(\frac{2\sigma_1^4\sigma_2^2}{2\sigma_1^4+\sigma_2^2r^2}\right). \tag{20}$$

4 Approaches to Sharper Constrained Bounds

The counterexample above illustrates some of the difficulties that can arise in forming a local lower bound as the limit of a sequence of lower bounds computed over test points converging to $\underline{\theta}$. In particular, it is not possible to exchange the limit and expectation operations in (4) because of the discontinuity in $J_{\underline{\theta}}^{+}(\Delta)$ in the limit. The problem briefly alluded to in Section 2 is that the eigenvalues associated with eigenvectors of $J_{\underline{\theta}}^{+}(\Delta)$ lying in the hyperplane normal to the constraint surface $\Theta_C = \{\underline{\theta} \in \mathbb{R}^n : G_{\underline{\theta}} = 0\}$ approach infinity at the rate $O(\Delta^2)$. In the counterexample above, the finite-difference vector, $\delta \underline{m}_{\underline{\theta}}$ has components approaching zero at the rate $O(\Delta)$ which lie in the eigenspace associated with these unbounded eigenvalues. Taking the limit of the quadratic form as in (4), we obtain a bound which is sharper than the constrained CR bound. It is interesting to note that the second term in (20) is a correction term which depends upon:

(i) the curvature r of the constraint surface Θ_C,

(ii) the Fisher information σ_2^2 along the normal (e.g., along $\underline{e}_2$) to the constraint surface, and

(iii) higher-order Fisher information σ_1^4 along the tangent plane.

This is in contrast to the constrained CR bound which depends only on the first-order Fisher information σ_1^2 along the tangent plane.

Intuition might lead one to suspect that a higher-order Bhattacharyya-type constrained lower bound exists. Indeed, this is the case, although we do not have sufficient space to develop the bound in full detail here. For the case of a constraint set Θ_C defined by a functional relationship, the constrained-parameter analog to the Bhattacharyya bound is constructed by projecting the higher-order derivatives of the likelihood function onto the space of higher-order derivatives of the constraint function. Consider the second-order Bhattacharyya Bound. Define the partial-derivative operator $\partial_i \stackrel{\text{def}}{=} \frac{\partial}{\partial \theta_i}$. Further, assume that the second partial derivatives of the likelihood function are square-integrable. Then define:

$$\tilde{M}_{\underline{\theta}} = \left[[\partial_i \underline{m_{\theta}}]_i \;\; [\partial_j \partial_k \underline{m_{\theta}}]_{jk} \right], \tag{21}$$

the lexicographically-ordered matrix of first and second partial derivatives of $\underline{m_{\theta}}$. Similarly, define the vector:

$$\underline{\zeta_{\theta}} \stackrel{\text{def}}{=} \left[[\frac{\partial_i p_{\underline{\theta}}}{p_{\underline{\theta}}}]_i \;\; [\frac{\partial_j \partial_k p_{\underline{\theta}}}{p_{\underline{\theta}}}]_{jk} \right], \tag{22}$$

and the matrix:

$$\tilde{J}_{\underline{\theta}} = \mathrm{E}_{\underline{\theta}}\{\underline{\zeta}_{\underline{\theta}}^T \underline{\zeta}_{\underline{\theta}}\}. \tag{23}$$

Then the standard second-order Bhattacharyya bound is [10, page 148]:

$$\Sigma_{\underline{\theta}} \geq \tilde{M}_{\underline{\theta}} \tilde{J}_{\underline{\theta}}^{-1} \, \tilde{M}_{\underline{\theta}}^T. \tag{24}$$

Analogous to the bound of (5), we have the constrained Bhattacharyya bound:

$$\Sigma_{\underline{\theta}} \geq \tilde{M}_{\underline{\theta}} \tilde{Q}_{\underline{\theta}} \tilde{J}_{\underline{\theta}}^{-1} \tilde{M}_{\underline{\theta}}. \tag{25}$$

As in the constrained CR bound, the matrix $\tilde{Q}_{\underline{\theta}}$ is an oblique projector onto the product space of all gradient vectors (e.g., first-order partials) and hessian matrices (e.g., second-order partials) associated with the constraint set Θ_C. Thus the second-order constrained Bhattacharyya bound is a function of both the tangent planes and curvature of the constraint. In the normal circle model above, the constrained second-order Bhattacharyya bound on the variance of constrained unbiased estimators of $\tau_{\underline{\theta}} = \underline{\theta}^T \underline{\theta}$ is identical to the result (20) we obtained by taking the limit of the Barankin bound (2) over curved paths.

References

[1] J. Aitchison and S. D. Silvey, "Maximum-Likelihood Estimation of Parameters Subject to Restraints," Ann. Math. Stat., Vol. 29, pp. 813-828, 1958.

[2] S.-I. Amari, "Differential Geometry of Curved Exponential Families—Curvatures and Information Loss," *Ann. Stat.,* vol. 10, pp, 357-385, 1982.

[3] E. B. Barankin, "Locally-best unbiased estimates," *Ann. Math. Stat.*, vol. 20, pp. 477-501, 1949.

[4] J. M. Begun, W. J. Hall, W-M Huang, and J.A. Wellner, "Information and asymptotic efficiency in parametric-nonparametric models," *Ann. Statist.*, vol. 11, no. 2, pp. 432-452, 1983.

[5] D. G. Chapman and H. Robbins, "Minimum variance estimation without regularity assumptions," *Ann. Math. Statist*, vol. 22, pp. 581-586, 1951

[6] J. D. Gorman and A. O. Hero, "Lower Bounds for Parametric Estimation with Constraints," *IEEE Trans. on Info. Theory,* to appear in November 1990.

[7] J. Kiefer, "On Minimum Variance Estimators," *Ann. Math. Stat.*, vol. 23, no. 4, pp. 75-77, December, 1952.

[8] J. Pfanzagl and W. Wefelmeyer, *Contributions to a General Asymptotic Statistical Theory*, Lecture Notes in Statistics, vol. 13, Springer-Verlag, New York, 1982.

[9] G. W. Stewart, "On the continuity of the generalized inverse," SIAM J. Appl. Math., vol. 17, no. 1, pp. 33-45, Jan. 1969.

[10] H. L. VanTrees, *Detection, Estimation, and Modulation Theory, Part I*, J. Wiley and Sons, New York, 1968.

ON THE RELATION OF CROSS-VALIDATION AND ALTERNATIVE CRITERIA TO ESTIMATE THE REGULARIZATION PARAMETER

Nikolas P. Galatsanos[1] and *Aggelos K. Katsaggelos*[2]

[1] Electrical and Computer Engineering
Illinois Institute of Technology
Chicago, Illinois 60616.

[2] Electrical Engineering and Computer Science
Northwestern University
Evanston, Illinois 60208.

In a number of image processing problems ill-conditioned equations must be solved. A representative example is the problem of image restoration [2]. Regularization [5-7] is an effective method for obtaining satisfactory solutions to such problems. In many practical situations the image degradation can be adequately modeled by a linear shift-invariant blur and an additive white Gaussian noise. Then the degradation model is

$$g = H f + n, \tag{1}$$

where vectors g, f, n are M^2 lexicographic orders of the degraded observed image, the original image and the noise, respectively. The degradation matrix H is an $M^2 \times M^2$ block Toeplitz matrix and is ill-conditioned.

Cross-validation has been studied by statisticians in the context of ill-posed problems for a long time (see for example [8] and [3]). However, only recently this mathematical tool has attracted attention from the image processing community. Very recently Reeves and Mersereau [7] used cross-validation to estimate the regularization parameter λ in the image restoration problem. According to this method the regularization parameter can be found without knowledge of the noise variance. In [7] it is shown experimentally that using cross-validation yields a better estimate of the regularization parameter than using the residual equation as in [4]. The same observation was made by Craven and Wahba [1] in a very similar regularization problem, that of fitting noisy data with splines.

If the equation for $\hat{f}$ in [4] is used, the ordinary cross-validation function is defined by

$$P(\lambda) = ||B\ (A(\lambda) - I)\ g\,||^2, \tag{2}$$

where

$$A(\lambda) = H\ (H^t H + \lambda Q^t Q\ )^{-1} H^t, \tag{3}$$

and B is an $M^2 \times M^2$ diagonal matrix defined by

$$[B\,]_{jj} = 1/(1 - [A(\lambda)]_{jj}) \tag{4}$$

with $[A(\lambda)]_{jj}$ the diagonal elements of $A(\lambda)$. The minimization of $P(\lambda)$ with respect to λ provides an estimate of the regularization parameter.

In this paper the distance measure $T(\lambda)$ between the original image f and the estimated $\hat{f}$ is given by

$$T(\lambda) = ||Hf - H\hat{f}\,||^2 = ||f - \hat{f}\,||_H^{\,2} \tag{5}$$

is used, where $||\ \ ||_H$ denotes a weighted norm. If the noise n is white with zero mean and variance σ^2 we have showed that

$$E[\ \|Hf - H\hat{f}\|^2\ ] = \|\ (I - A(\lambda))g\ \|^2 + \sigma^2[2\cdot trace[\ A(\lambda)\ ] - M^2]. \qquad (6)$$

where E is the expectation operator. The implication of this result is that an estimate of the regularization parameter can be also computed by minimizing equation (6), if σ is known. Using this measure implies that data corresponding to the larger eigenvalues of H are weighted more. We have also shown that the value of λ that minimizes equation (6) satisfies the following equation

$$E\ \|\ A(\lambda)^{1/2}(A(\lambda) - I)\ g\ \|^2 - \sigma^2\cdot trace[\ A(\lambda)(I - A(\lambda))] = 0. \qquad (7)$$

Golub et al. showed in [3] that the value of λ that minimizes $E[\ P(\lambda)\ ]$ also minimizes $E[\ T(\lambda)\ ]$. Therefore using Golub's result and our our previous findings (Eqs. (6) and (7)), we propose using equation (7) to estimate the noise variance. In this paper we have further shown that for a particular choice of the regularizing operator this value of the regularization parameter λ that minimizes $E[\ P(\lambda)\ ]$, $E[\ T(\lambda)\ ]$ and satisfies equation (7) is equal to the noise variance.

Experiments have been performed using degraded images with shift invariant blurs and additive white noise. We verified that cross-validation, minimization of $E[\ T(\lambda)\ ]$ (see Eq. (6)), and equation (7) yield identical values for the regularization parameter for a number of blurs, signal-to-noise ratios and regularizing operators. The proposed noise estimator was also tested; it was found that it yields very accurate estimates of the noise variance.

REFERENCES

[1] P. Craven and G. Wahba, "Smoothing Noisy Data with Spline Functions," *Numerische Mathematik*, Vol. 31, pp. 377-403, 1975.

[2] G. Demoment, "Image Reconstruction and Restoration: Overview of Common Estimation Problems," *IEEE Trans. Acoustics Speech and Signal Processing*, Vol. 37, No. 12, pp. 2024-2036, December 1989.

[3] G. H. Golub, M. Hearth, and G. Wahba, "Generalized cross-validation as a method for choosing a good ridge parameter", *Technometrics*, Vol. 21, No. 2, pp. 215-233, 1979.

[4] B. R. Hunt, "The application of constrained least squares estimation to image restoration by digital computer," *IEEE Trans. Computers*, C-22, pp. 805-812, September 1973.

[5] A. K. Katsaggelos, "Iterative image restoration algorithms," *SPIE, Optical Engineering*, Vol. 28, no. 7, pp. 735-748, July 1989.

[6] A. K. Katsaggelos, J. Biemond, R. W. Schafer, and R. M. Mersereau "A regularized iterative image restoration algorithm", *IEEE Trans. Acoustic, Speech, and Signal Processing*, to appear, April 1991.

[7] S. J. Reeves and R. M. Mersereau, "Optimal estimation of the regularization parameters and stabilizing functional for regularized image restoration", *SPIE, Optical Engineering*, Vol. 29, No. 5, pp. 446-454, May 1990.

[8] G. Wahba, "Practical approximate solutions to linear operator equations when the data are noisy," *SIAM, Journal Numer. Anal.*, Vol. 14, No. 4, pp. 651-667, September 1977.

Estimation of NonGaussian Signals Using a Modified Gaussian Sum Adaptive Filter

Mauro J. Caputi, Richard L. Moose, William T. Baumann
Virginia Polytechnic Institute & State University
The Bradley Department of Electrical Engineering
Blacksburg, Virginia 24061

Abstract

The Gaussian sum estimation method developed by Sorenson and Alspach [1] is investigated. An alternate development of the Gaussian sum density approximation is given and a modified adaptive estimation structure is proposed based on the adaptive Kalman filter scheme first presented by Magill [2], and extended by Moose and Wang [3]. A necessary condition for effective estimation is examined [5]. Several alternate configurations are proposed when this condition is not met. Simulation results illustrating the theory are analyzed. A suboptimal method of tuning the parameters of the adaptive structure is suggested.

Introduction

An estimation technique, applicable to both linear systems with nonGaussian inputs and nonlinear systems with Gaussian inputs, has been developed by Sorenson and Alspach [1]. In the case of linear systems, the noise processes associated with the plant and measurement are assumed known and nonGaussian. The probability density function of each noise process is approximated by a Gaussian sum; that is, a weighted sum of Gaussian density functions. The Gaussian sum approximation is written as

$$p_{GS}(x) = \sum_{i=1}^{M} \alpha_i \mathrm{N}[\mu_i, \sigma_i^2] \tag{1}$$

where

$$\sum_{i=1}^{M} \alpha_i = 1; \quad \alpha_i \geq 0 \ \text{ for } i = 1, 2, \ldots, M \tag{2}$$

and

$$\mathrm{N}[\mu_i, \sigma_i^2] = \frac{1}{\sqrt{2\pi}\sigma_i} \, e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma_i}\right)^2} \tag{3}$$

This work is supported by the Office of Naval Research, grant N00014-89-J-3123.

For sufficiently large M, any density function can be closely approximated by a Gaussian sum. As long as condition (2) holds, the Gaussian sum is always a valid density function.

Alternate Gaussian Sum Development

In developing their estimator, Sorenson and Alspach begin by directly representing the plant and measurement noise processes by Gaussian sums. The conditional density $p(x_k \mid Z_k)$ of the current state vector x_k given the current measurement sequence $Z_k = \{z_1, z_2, \ldots, z_k\}$ is then updated using Bayes' rule.

A major problem with their development is that the number of terms in the Gaussian sum increases at each iteration of their estimator. This problem of *growing memory* can be avoided by the following alternate development of the Gaussian sum density approximation. The nonGaussian noise process or input signal is modeled as the sum of two statistically independent random processes

$$u = b + n \tag{4}$$

The first term, b, is a semi-Markov process with state transitions governed by the transition probability matrix of a conventional Markov process. However, the amount of time spent in state i before the next transition to state j is a random variable [4]. Its values lie within a *fixed set* of discrete values, characterized by a delta probability density function

$$p(b) = \sum_{i=1}^{M} P_i \delta(b - b_i) \tag{5}$$

with

$$\sum_{i=1}^{M} P_i = 1; \quad P_i \geq 0 \text{ for } i = 1, 2, \ldots, M \tag{6}$$

This process can be thought of as a randomly-switching bias, each bias value b_i having probability P_i. The second term, n, is a zero mean Gaussian process with variance σ_i^2. The variance switches at the same time the bias switches. This switching variance can be thought of as changing the power of the Gaussian process.

With both densities known, and using the convolution relationship between u, n, and b, the density function of u is given by

$$p_{GS}(u) = \sum_{i=1}^{M} P_i \mathrm{N}[b_i, \sigma_i^2] \tag{7}$$

Thus, the nonGaussian density function of u can be modeled as a Gaussian sum. The weight of each Gaussian term is the probability of the i^{th} biasing term. By allowing only a fixed number of bias values, the number of terms in the Gaussian sum at each iteration of the estimator is fixed, thereby eliminating the growing memory problem.

Selecting the parameters P_i, b_i, and σ_i in (7) to obtain the "best" approximation p_{GS} to some actual nonGaussian density function p is accomplished by means of minimizing the L^k norm

$$\| p - p_{GS} \|^k = \int_{-\infty}^{\infty} \left| p(u) - \sum_{i=1}^{M} P_i \mathrm{N}[b_i, \sigma_i^2] \right|^k du \tag{8}$$

This curve fitting exercise can be done off-line using several values of M until a suitable trade-off between minimum norm and minimum M is obtained.

Development of a Modified Gaussian Sum Adaptive Filter

A modified Gaussian sum adaptive filter is now developed for a general linear system with a nonGaussian input signal and Gaussian plant and measurement noise processes. The system is modeled in standard discrete-time state-space form as

$$x_{k+1} = \Phi x_k + \Gamma u_k + \Psi w_k \tag{9}$$
$$z_k = H x_k + D u_k + v_k \tag{10}$$
$$u_k = b_k + n_k \tag{11}$$

where		
	x	is the state vector
	z	is the measurement vector
	w	is a zero mean white Gaussian plant noise process with covariance Q
	v	is a zero mean white Gaussian measurement noise process with covariance R, independent of w
	Φ, Γ, Ψ, H, D	are the respective constant transition matrices
	u	is the vector Gaussian sum signal model (4) of the actual nonGaussian input signal, comprised of semi-Markov bias vector b_k, and zero mean white Gaussian noise n_k with covariance S_k

The optimal estimate of the state vector is found from the conditional mean as

$$\hat{x}_{k+1} = E[x_{k+1} \mid Z_{k+1}] = \int_{-\infty}^{\infty} x_{k+1}\, p(x_{k+1} \mid Z_{k+1})\, dx_{k+1} \tag{12}$$

where Z_{k+1} is the measurement sequence $\{z_1, z_2, \ldots, z_{k+1}\}$. Using Bayes' theorem, the conditional density function of (12) can be written as

$$p(x_{k+1} \mid Z_{k+1}) = \frac{p(x_{k+1}, Z_{k+1})}{p(Z_{k+1})} \tag{13}$$

The bias vector b_k and covariance matrix S_k are explicitly brought into (13) by considering the joint density $p(x_{k+1}, Z_{k+1})$ to be a marginal density found from $p(x_{k+1}, Z_{k+1}, b_k, S_k)$ by summing over the b_k and S_k terms

$$p(x_{k+1} \mid Z_{k+1}) = \frac{\sum_{i=1}^{M} p(x_{k+1}, Z_{k+1}, b_i, S_i)}{p(Z_{k+1})} \tag{14}$$

After some additional algebraic manipulation, the optimal estimate of (12) becomes

$$\hat{x}_{k+1} = \sum_{i=1}^{M} \hat{x}^i_{k+1}\ p(b_i, S_i \mid Z_{k+1}) \tag{15}$$

where $\hat{x}^i_{k+1}$ is the conditional mean estimate of x_{k+1} given that $b_k = b_i$ and $S_k = S_i$, denoted by

$$\hat{x}^i_{k+1} = \int_{-\infty}^{\infty} x_{k+1}\ p(x_{k+1} \mid Z_{k+1}, b_i, S_i)\ dx_{k+1} \tag{16}$$

and the weighting factor $p(b_i, S_i \mid Z_{k+1})$ is the probability that $b_k = b_i$ and $S_k = S_i$ given the current measurement sequence. The overall estimate of (15) is then a weighted sum of individual estimates, each based on a particular set of parameters b_i and S_i.

Each estimate (16) is found by a modified Kalman filter. Inserting (11) into (9) and (10) produces

$$x_{k+1} = \Phi x_k + \Gamma b_k + w'_k \tag{17}$$

$$z_k = H x_k + D b_k + v'_k \tag{18}$$

where

$$w'_k = \Gamma n_k + \Psi w_k \tag{19}$$

$$v'_k = D n_k + v_k \tag{20}$$

Letting w_k, v_k, and n_k be statistically independent zero mean white Gaussian processes, and recognizing that $E[w'_{k-1} v'_k] = 0$, the Kalman filter equation for (15) is

$$\hat{x}^i_{k+1} = \Phi \hat{x}^i_k + \Gamma b_i + K^i_{k+1}[z_{k+1} - D b_i - H(\Phi \hat{x}^i_k + \Gamma b_i)] \tag{21}$$

with covariance and gain equations

$$M^i_{k+1} = \Phi P^i_k \Phi^{\mathrm{T}} + \Gamma S_i \Gamma^{\mathrm{T}} + \Psi Q \Psi^{\mathrm{T}} \tag{22}$$

$$K^i_{k+1} = M^i_{k+1} H^{\mathrm{T}} (H M^i_{k+1} H^{\mathrm{T}} + D S_i D^{\mathrm{T}} + R)^{-1} \tag{23}$$

$$P^i_{k+1} = (I - K^i_{k+1} H) M^i_{k+1} \tag{24}$$

The structure of the overall adaptive filter is a bank of Kalman filters operating in parallel, with each individual estimate multiplied by its own weighting term probability. This is the basic structure as outlined in Magill [2], with some modifications as will be shown. The filter based on the parameter set that most closely matches the actual parameters of the modeled system will have a weighting term that tends closer to one, while the weights of the mismatched filters will tend towards zero.

Using Bayes' rule and writing Z_{k+1} as $\{Z_k, z_{k+1}\}$, the weighting term becomes

$$p(b_i, S_i \mid Z_{k+1}) = \frac{p(z_{k+1} \mid b_i, S_i, Z_k)\ p(b_i, S_i \mid Z_k)}{p(z_{k+1} \mid Z_k)} \tag{25}$$

The denominator term is independent of i and is a scale factor that ensures that the sum of the weights (25) at each iteration is equal to one. The second term of the numerator can be rewritten to explicitly include the sample time for the bias and covariance terms

$$p(b_i, S_i \mid Z_k) = p(b_{k+1} = b_i, S_{k+1} = S_i \mid Z_k) \tag{26}$$

Using Bayes' rule and the definition of marginal densities, (26) can be written as

$$p(b_i, S_i \mid Z_k) = \sum_{j=1}^{M} p(b_{k+1} = b_i, S_{k+1} = S_i \mid b_k = b_j, S_k = S_j)\, p(b_k = b_j, S_k = S_j \mid Z_k) \tag{27}$$

The following notation will be used to express the two terms of the summation (27)

$$\Theta_{i,j} = p(b_{k+1} = b_i, S_{k+1} = S_i \mid b_k = b_j, S_k = S_j) \tag{28}$$

$$w_k^j = p(b_k = b_j, S_k = S_j \mid Z_k) \tag{29}$$

The density function of (28) is a Markov transitional probability matrix [4]; that is, $\Theta_{i,j}$ is the conditional probability that $b = b_i$ and $S = S_i$ at time $k+1$, given that $b = b_j$ and $S = S_j$ at time k. The density function of (29) is of the same form as (25) and is just the previous weighting term at the previous value of time k.

Necessary Condition for Effective Estimation

The first term of the numerator of (25) is of prime importance in determining whether or not the adaptive filter will work properly for a given system. It can be modeled as a Gaussian density if the bias and covariance terms switch slowly compared to the sample interval k. This assumption will be made here and has been verified by extensive simulation and analysis by Moose and Wang [3]. This conditional density is

$$p(z_{k+1} \mid b_i, S_i, Z_k) = \mathrm{N}[H(\Phi\hat{x}_k^i + \Gamma b_i) + Db_i,\ HM_{k+1}^i H^{\mathrm{T}} + DS_i D^{\mathrm{T}} + R] \tag{30}$$

The i^{th} measurement residual from (21) is

$$r_{k+1}^i = z_{k+1} - Db_i - H(\Phi\hat{x}_k^i + \Gamma b_i) \tag{31}$$

The mean and covariance of (30) are readily available from the residual (31) and the Kalman gain (23). Essentially, the measurement residual of the matched filter will have the smallest expected value (typically, zero mean), while the residuals of all the mismatched filters will be biased. Under the Gaussian assumption, the probability of the matched filter will be the largest among all the filters.

For some systems, however, the statistical steady-state value (that is, the steady-state value of the expected value) of the measurement residual of *each* filter goes to zero. None of the residuals are biased. Therefore, the adaptive system cannot detect which filter has the correct parameter set and the weights all tend to the same value.

The statistical steady-state (SSS) value of the measurement residual is a function of the dc gain of the system transfer function [5]. If the dc gain of the system is zero, then the SSS value of the residual is zero. This condition provides a simple test as to whether or not this adaptive system will work properly for a particular system.

Alternate Estimation Procedures

A possible solution to this problem is to first determine if an alternate measurement provides a nonzero dc gain. For example, if in the zero dc gain system acceleration measurements are taken, changing to velocity measurements may provide a nonzero dc gain. If the actual sensors producing these measurements cannot be changed, integrating the acceleration measurement data to produce *approximate*-velocity measurement data may allow the adaptive system to work.

To illustrate this point, consider a general modal structure containing modes which are closely spaced in frequency and heavily damped. The input to the system is a nonGaussian stochastic FM signal, not directly measurable by the system. The model of this system follows (9-11). Checking the dc gain of this system using an acceleration measurement (with no noise)

$$G(\omega)\Big|_{\omega=0} = G_0 = H(I-\Phi)^{-1}\Gamma + D \tag{32}$$

gives us zero dc gain. Since the SSS value of the measurement residual is a function of this dc gain

$$r_{ss}^i = [I - H\Phi(I - (I + K_{ss}H)\Phi)^{-1}K_{ss}]G_0(u_k - b_i) \tag{33}$$

the residuals for all filters go to zero when the system reaches SSS. Therefore, the residual density function $p(z_{k+1} \mid b_i, S_i, Z_k)$ cannot discern which filter is the matched filter and all the weighting terms (25) become equal.

Changing this modal system to output velocity measurement data produces a transfer function with a nonzero dc gain. Typically, changing the actual system cannot be done in practice. However, given a large signal to noise ratio between the input signal and measurement noise, a good approximation of the velocity measurement data can be generated by integrating the acceleration measurement data from the original system using a first or second order discrete time integrator. Redesigning the residual density function to be based on the velocity measurement model, while actually using *approximate*-velocity measurement data, will cause the weighting terms to properly select the matched filter.

Simulation Analysis of Alternate Adaptive Filters

Several simulations were run to test the performance of the Gaussian sum filter under various conditions. One simulation implements the velocity measurement system with nonzero dc gain, and uses an adaptive filter design based on the velocity measurement model. This serves as a benchmark for comparison with the other simulations.

A second simulation implements the acceleration measurement system with zero dc gain. Two adaptive filters are used. One filter is designed based on the acceleration measurement model, and produces the individual state estimates from (21). The second filter is designed based on a velocity measurement model, but uses *approximate*-velocity measurement data generated from a discrete time integrator. This filter is used to produce the proper weighting terms (25). The overall performance is not as good as the filter of the first simulation. The second filter could also be used to generate the individual state estimates, but the performance is no better than the first filter since the same acceleration measurement data is used to generate the *approximate*-velocity measurement data.

A third simulation implements a dual measurement system, where the measurement is comprised of two models: the acceleration measurement model, and the velocity measurement model. The adaptive filter is designed based on the dual measurement model and uses acceleration measurement data plus *approximate*-velocity measurement data. This filter can be tuned to perform better than the filter of the second simulation, but not as well as the filter of the first simulation.

Determination of Approximate Parameter Values

As mentioned previously, the system cannot directly measure the input signal. Referring to the stochastic FM input, the bias parameters b_i used to model its density as a Gaussian sum are based on the amplitude of the signal. For good performance, the adaptive filter must have some approximate knowledge of this amplitude.

One possible method of determining amplitude information [5] is to monitor and process the measurement residual of the zero-bias filter (the individual filter with a bias parameter value of zero) of the weighting term adaptive filter of the second simulation described previously. Setting $b_i\big|_{i=0} = 0$ in (31) gives

$$r^0_{k+1} = z_{k+1} - H\Phi\hat{x}^0_k \tag{34}$$

The velocity measurement model, given below, of the weighting term adaptive filter is of the same form as (10), except $D = 0$

$$z_{k+1} = Hx_{k+1} + v_{k+1} \tag{35}$$

Using (9), (35), and setting the error $e^0_k = (x_k - x^0_k)$, the zero-bias residual (34) becomes

$$r^0_{k+1} = H\Phi e^0_k + H\Gamma u_k + H\Psi w_k + v_{k+1} \tag{36}$$

Given that the input signal u_k is much larger than both w_k and v_{k+1}, and then premultiplying both sides of (36) by $(H\Gamma)^{-1}$, the zero-bias residual is processed as

$$\tilde{u}_{k+1} = (H\Gamma)^{-1} r^0_{k+1} \simeq (H\Gamma)^{-1} H\Phi e^0_k + u_k \tag{37}$$

If the expected value of the first term on the right hand side of (37) is close to zero, then the expected value of the processed zero-bias residual can be approximated by

$$\mathrm{E}[\tilde{u}_{k+1}] \simeq u_k \tag{38}$$

To approximate this expected value, a smoothed version of $\tilde{u}_{k+1}$ is found from a first order discrete time smoother

$$\tilde{u}^s_{k+1} = \alpha_s \tilde{u}^s_k + (1 - \alpha_s)\tilde{u}_k \tag{39}$$

The peak values and power of an n-sample window of (39) are calculated and used to periodically update the bias parameters in the remaining nonzero-bias filters in the adaptive filter bank. This could serve as a suboptimal method of tuning the bias parameters to their "optimal" values when the input signal amplitude is originally unknown, or if the signal makes large changes in amplitude at unknown times.

References

[1] H. W. Sorenson and D. L. Alspach, "Recursive Bayesian Estimation Using Gaussian Sums," *Automatica*, vol. 7, pp. 465-479, 1971.

[2] D. T. Magill, "Optimal Adaptive Estimation of Sampled Stochastic Processes," *IEEE Trans. Auto. Cont.*, vol. AC-10, no. 4, Oct. 1965.

[3] R. L. Moose and P. P. Wang, "An Adaptive Estimator with Learning for a Plant Containing Semi-Markov Switching Parameters," *IEEE Trans. on Sys., Man, and Cyber.*, vol. SMC-3, no. 3, May 1973.

[4] R. A. Howard, *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes.* New York: Wiley, 1971.

[5] M. J. Caputi, "NonGaussian estimation using a modified Gaussian sum adaptive filter," Ph.D. dissertation in preparation, Virginia Polytech. Inst. & S. U., Blacksburg, Virginia.

Performance of an Adaptive Detection Scheme in an Impulsive Correlated Environment

Pankaj Srivastava and Peter Willett *
University of Connecticut, Storrs, CT 06269

It is well-known that the performance of the matched filter degrades markedly when the ambient Gaussian noise process is corrupted by groups of extraneous noise samples. Here we propose a scheme whereby such *bursts* are with high probability recognized, and their effects minimized.

It is convenient [1] to express the matched filter in terms of the ambient noise innovations (or prediction-errors) as

$$T(\vec{x}) = \sum_{n=1}^{N} \frac{g_{m(n)}(n) f_{m(n)}(n)}{\sigma^2{}_{m(n)}}$$

where N is the length of the detection record; $f_{m(n)}(n)$ is the $m(n)^{th}$ order prediction error (PE) of variance $\sigma^2{}_{m(n)}$ at time "n"; and $g_{m(n)}(n)$ denotes the corresponding PE filter output with the signal as input. The PE order $m(n)$ is simply $n-1$ for the standard matched filter, and may be considered to be clamped at p if the ambient noise process is autoregressive (AR) of that order. Note that the lattice implementation of the PE filter allows easy access to all prediction errors.

The unpredictability associated with a burst is magnified in the process $\{f_m(n)\}$, and we may use this to recognize a burst's onset. A scheme which simply ignores observations with large (greater in magnitude than $4\sigma_{m(n)}$) prediction errors, which we call the "matched-filter-blanker" (MFB), suffers from the drawback that it must in general wait a considerable time following a burst for small PE's to appear so it can re-start. This seems true regardless of whether or not the offending samples are deleted from the PE filter.

We propose to use a "start-over" (SO) rule for the predictor order. That is, $m(n)$ is increased by one for each n, unless $|f_{m(n)}| > \tau_{m(n)}\sigma_{m(n)}$ under *both* hypotheses, in which case $m(n+1)$ is set to zero and the present sample ignored. We have found $\tau_0 = \tau_1 = 2$ and $\tau_m = 4$ for $m \geq 2$ to be a robust choice of thresholds, with relatively few false bursts, and with spurious re-starts having little effect. The operation is illustrated in Figure 1, where we have represented a deleted sample by a "negative" predictor order.

Figures 2 through 6 show the effects of various parameters on performance. We have chosen as ambient noise the 9^{th}-order AR process found in [2] to be representative of certain underwater acoustic data; specifically, we have

$$\vec{a} \approx \{1, -4.15, 9.4, -14.8, 17.7, -16.3, 11.6, -6.19, 2.25, -.43\}$$

Each figure is the result of 10^6 simulated detection records, and unless otherwise stated a record-length of 20; an average burst-length (uniformly-distributed between unity and twice that number) of 5; a burst-to-ambient noise ratio of $5dB$, and bursts occurring on average once every 10 records. The SNR is $3dB$ for the constant signal cases and $-27db$ for the alternating signal, of which both values are varied inversely with the record length in Figures 5 and 6.

We note that in almost all cases the matched filter performs very poorly, even for "small" bursts. The SO detector is considerably better than the MFB in all situations except that of Figure 6. In this case, heuristically, detection cannot "begin" until the predictor order is at its maximum, and hence the MFB and SO detector are essentially identical.

References

[1] P.Srivastava and P.Willett, "Detection in Impulsive Correlated Environments", *Proc. of the Conference on Information Sciences and Systems*, Princeton University, March 1990.

[2] P. Srivastava and P. Willett, "An Analysis of Non-Stationary Under-Ice Noise", *Proc. of the Conference on Information Sciences and Systems*, March 1989.

---

*Supported by the National Science Foundation under Grant ECS-8719031.

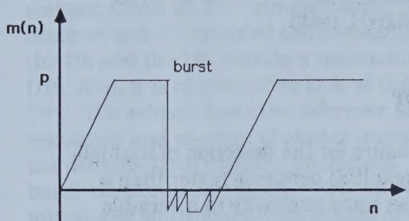


Figure (1) – Predictor Order

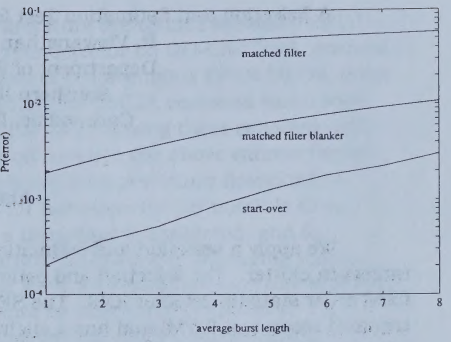


Figure (4) – Constant Signal

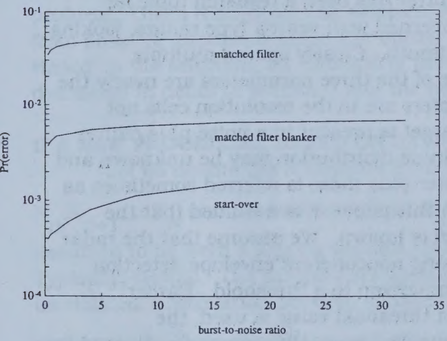


Figure (2) – Constant Signal

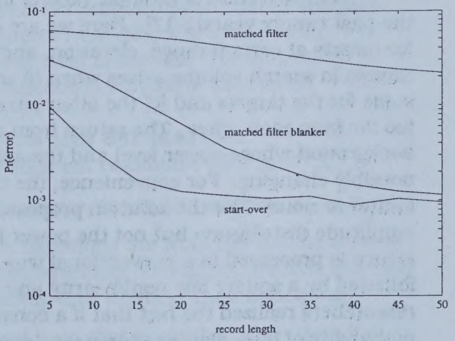


Figure (5) – Constant Signal

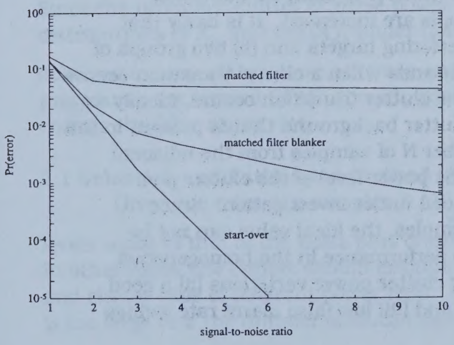


Figure (3) – Constant Signal

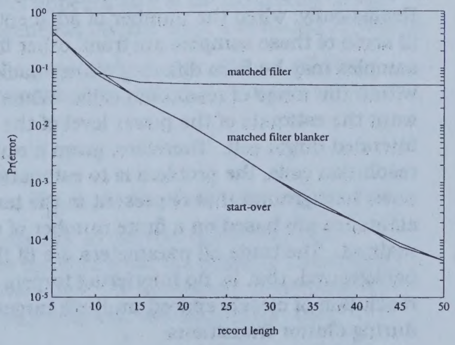


Figure (6) – Alternating Signal

A Selection and Estimation Test for Multiple Targets in Clutter Detection

R. Viswanathan and Abbas Eftekhari
Department of Electrical Engineering
Southern Illinois University
Carbondale, Illinois 62901-6603

ABSTRACT

We apply a selection and estimation procedure for the detection of multiple targets in clutter. The selection and estimation test (SE) performs better than a fixed order statistic detector (OS). The SE test has some similarity to a variable trimmed mean test (VTM) and has a slight performance advantage. Unlike the VTM, whose performance is evaluated only by simulation studies, the performance of the SE test, for Swerling type I target model, can be evaluated analytically. Further improvement in the performance is possible if a better selection procedure can be found.

I INTRODUCTION

The detection of multiple targets in clutter has been a research topic for the past twenty years[1-17]. Here we are concerned with search type radars, looking for targets at certain range, elevation, and azimuth. Closely spaced multiple targets in search volume arises when (i) some of the three parameters are nearly the same for the targets and (ii) the other parameters are in the resolution cells not too far from each other. The return from a target is present in a noise plus clutter background whose power level and the amplitude distribution may be unknown and possibly changing. For convenience, the clutter plus noise is referred sometimes as clutter or noise. For the solution proposed in this paper, it is assumed that the amplitude distribution but not the power level is known. We assume that the radar return is processed in a conventional way, using noncoherent envelope detection followed by a square law nonlinearity and comparison to a threshold. Earlier researchers realized the fact that if a constant threshold value is used, the probability of false alarm can increase dramatically, even when the power level of the clutter changes by a small amount. In target detection, it is important to keep the probability of false alarm below a specified value. Therefore, it becomes necessary to take samples from adjacent resolution cells in order to have an estimate of the clutter plus noise level. If the resolution cells are purely from a clutter and noise background, one can have a consistent estimator whose value will approach asymptotically the clutter power level, as the number of samples increases. Realistically, when the number of adjacent cells are increased, it is likely that (i) some of these samples are from other interfering targets and (ii) two groups of samples may be from differing clutter backgrounds when a clutter transition occurs within the range of resolution cells. When the clutter transition occurs, ideally we want the estimate of the power level of the clutter background that is present in the intended target cell. Therefore, given a number N of samples from the adjacent resolution cells, the problem is to estimate the power level of the clutter plus noise background that is present in the test cell under investigation. Since estimates are based on a finite number of samples, the ideal value can not be realized. The trade off parameters are (i) the performance in the homogeneous background, that is, no interfering targets or clutter power variations (ii) a good resolution of closely spaced multiple targets and (iii) low false alarm rate swings during clutter transitions.

A number of schemes proposed in the literature are termed variously as (1) CA-CFAR, cell averaging constant false alarm rate detector (2) GO-CA-CFAR, greatest of cell averaging CFAR (3) SO-CA-CFAR, smallest of cell averaging CFAR (4) OS, order statistic CFAR (5) TM, trimmed mean CFAR [10,18] (6) CMLD, censored mean level detector and (7) weighted cell averaging CFAR [10,14]. Among these schemes, only the OS and the TM provide a reasonable trade off between the above various factors [10]. Also, it is of interest to look at the multiple decision procedure described in [9]. The scheme tests the reference samples for homogeneity, for possible clutter transition and position of clutter transition if a transition is suspected, and for the samples from possible interferers. The author has quoted encouraging results based on simulation studies. The drawback of such a multistage procedure is that it introduces statistical dependencies and it cannot be analytically evaluated. Simulation studies will have to be restricted to fewer situations. Other recent procedures are a variable trimmed mean test (VTM)suggested in [16] and an adaptive order statistic (AOS) test in [25]. Because of analytical difficulty one resorts to simulation for performance studies. The VTM performs slightly better than the OS detector. For few example cases, we compare the performances of the OS and the VTM to the selection and estimation (SE) test. The idea of selection and ranking used in statistics for reliability and other applications is applied to this estimation problem. Given N adjacent cell samples, a selection procedure determines samples from possible interfering targets or clutter samples from differing background. Based on the selection, an estimate of the power level is derived. The scheme is discussed in detail in section II and the performance is studied in section III.

II A NEW PROCEDURE BASED ON SELECTION AND ESTIMATION:

Let us denote the N samples from the adjacent resolution cells as $X_1, X_2, \dots, X_n, X_{n+1}, \dots X_N$ and the test sample as X_0. For convenience, the samples $X_1, \dots X_n$ are assumed to be from a lagging window and the remaining are from the leading window. Typically, n=N/2. The samples are assumed independent, though in practice this may be violated. Correlation between samples usually lead to worse than anticipated performance. If we denote the estimate derived from these samples as λ^*, then the test for the target detection is based on the following:

$$\text{Declare target present if } X_0 > t\,\lambda^* \tag{2.1}$$

where t is some positive number. For any of the schemes, assuming a homogeneous background, t will be found to achieve a specified probability of false alarm, P_F. Since the unknown clutter or noise power level appears as a scale parameter in the distributions of X_0 and the $\{X_i\}$, under homogeneous background, a constant P_F will be achieved for any of the above schemes and the proposed scheme. Hence, all these procedures are termed CFAR, though interfering targets and/or clutter transitions can change the false alarm rate.

2.1 Selection and Estimation Method (SE):

Given the reference samples $X_1, X_2, \dots, X_N$, a subset of samples with power levels equal to that of the noise plus clutter present in the test cell is required. In otherwords, the subset should exclude the samples from the possible interferers and clutter cells of different power level. Obviously if the number of interferers is too large or if the number of clutter cells with different levels are large, then the selected subset size will be small and the estimate λ^* will not be good. If this happens, nothing can be done to improve, but this is not the problem of the selection scheme. In reliability and other applications, selection and ranking schemes are widely used[19-21].

Various problems in selection and ranking can then be specified as follows:

(1) Selecting the one best population.

(2) Selecting the m best populations for $m \geq 2$, (a) in an ordered manner or (b) in an unordered manner.

(3) Selecting a random number of populations, say, r, that includes the m best populations.

(4) Ordering all the k populations from best to worst(or vice versa).

A subset selection approach is applied to select a subset of the reference samples $X_1, X_2, \ldots, X_N$. Ideally, the following aspects of selection are required. (1) All $X_1, \ldots, X_N$ should be selected, if they are identically distributed(i.e., homogeneous background). (2) If there are multiple interfering targets, the samples due to these should not be selected but the rest should be included. (3) If clutter background with differing power levels exist, the samples whose power levels are the same as the one present in the test cell must be selected and the rest should not be selected. In practice, all these requirements cannot be met. Let us consider the following procedure:

include in the subset all those X_i satisfying $X_i \leq d\, X_{(b)}$ (2.2)

where $d \geq 1$, and $X_{(b)}$ is the bth smallest order statistic of the samples $\{X_i\}$. Let r denote the size of the selected subset and $P_s(r)$ be the corresponding probability of correctly selecting the subset. This probability is given by

$$P_s(r) = \Pr(X_{(r)} \leq d\, X_{(b)} < X_{(r+1)}), \quad r = b, b+1, \ldots, N-1$$
$$\Pr(X_{(N)} \leq d\, X_{(b)}), \quad r = N \tag{2.3}$$

For the homogeneous background, the joint distribution of $X_{(b)}$, $X_{(r)}$, and $X_{(r+1)}$ is known in terms of the distribution of $\{X_i\}$, and hence (2.3) can be evaluated [22, 23]. In the sequel, we assume that $\{X_i\}$ and X_0 are exponentially distributed (Swerling type I target). The probability density functions of the samples are assumed as follows:

Homogeneous background:

$$f(x) = \begin{cases} 1/\lambda\, e^{-x/\lambda} & x \geq 0 \\ 0 & \text{elsewhere} \end{cases} \tag{2.4}$$

For convenience, denote the above density function as $X \sim \exp(\lambda)$.

$$X_i \sim \exp(1) \quad i = 1, 2, \ldots, N. \tag{2.5}$$

Without any loss of generality, the mean of the exponential is taken to be one when the samples are from the homogeneous background.

For the test cell,

$$X_0 \sim \exp(1) \quad \text{for no target}$$
$$\sim \exp(1+SNR) \quad \text{for target} \tag{2.6}$$

Clutter transition:

Assuming the clutter transition occurs at the $(\ell+1)$th sample,

$$X_1, \ldots, X_\ell \sim \exp(1)$$
$$X_{\ell+1}, \ldots, X_N \sim \exp(1+CNR) \tag{2.7}$$

where CNR denotes the clutter power to noise (or low clutter) power ratio.

For the test cell,

$$X_0 \sim \exp(1) \text{ or } \exp(1+CNR) \quad (2.8)$$

for no target, according to whether or not the test sample is from noise background or clutter background respectively.

$$X_0 \sim \exp(1+SNR) \text{ or } \exp(1+CNR+SNR) \quad (2.9)$$

for target, according to whether or not the test sample is from noise background or clutter background respectively.

Interferers:

When some of the samples are from interfering target returns, these samples are assumed distributed exponentially with parameter 1+INR, where INR denotes the interfering target power to noise power ratio.

The selection probability (2.3), for homogeneous background is found to be as follows:

$$P_S(r) = \binom{N}{r} \binom{r}{b} b/(d-1) \sum_{i=0}^{b-1} \binom{b-1}{i} (-1)^i \beta(r-b+1,(r-b+1+i+d(N-r))/(d-1)) \quad (2.10)$$

Above, β (,) denotes the beta function.

Tables 1 and 2 show the selection probabilities for different values of d and b and for homogeneous, five interferers, and the clutter transition in the middle situations. (The selection probabilities for the interferer and the clutter transition situations are obtained from P (f,r) expression in 3.1 with t replaced by zero.) For any combination of b and d, the selection scheme is far from ideal. Therefore, as will be seen below, a judicious choice of estimator, based on the selection r and some prior knowledge of the expected maximum number of interferers, is required to achieve an optimum performance. In general, for a given b, the choice of d is based on the following considerations. A large value of d will provide a good selection probability under the homogeneous background but will not provide good selection when interferers are present. Especially when the INR is large, such as 20dB, a good selection must be able to identify the interferers with high probability. As will be seen later, with a suitably designed estimator based on selection, the performance in the homogeneous background is not changed significantly for a wide range of d values, except for very small values. Hence, it would be productive to choose a small d value that is consistent with the following. The d value should not be so small that (i) it gives a large probability of selection for r values close to N/2 ,when there exists an expected maximum number of interfering targets of reasonable strength, thereby making the selection scheme confuse between the clutter transition in the middle and the interfering target situation (ii) it fails to identify rather poorly the clutter transition occurring in the middle (iii) it gives very low selection probability for r values in the upper half, for the homogeneous case. Based on the above discussion, and the tables, a value of 5 for d is reasonable for b=8 and a value between 10 and 15 is reasonable for b=4. In the next section we show how one can design an estimator based on the selection and achieve some definite improvement in the performance over the OS detector.

III PERFORMANCE OF SELECTION AND ESTIMATION

As pointed out in the previous section, a reasonable value for d can be chosen for a given value of b. The choice of b can then be decided by evaluating the performances for different b's . In order to show that a definite improvement in performance over the fixed OS detector is possible, we consider the following case.

Consider N=24, false alarm probability of 10^{-6}, and the 20th order statistic detector[10]. With INR/SNR = 1, the OS detector can tolerate up to 4 interfering targets with the detection probability of the test target approaching 1 as a function of increasing signal power, can provide a detection probability of only 0.27 with infinite test signal power, when there are 5 interferers. The performance in the homogeneous background and the false alarm swings with differing clutter backgrounds are shown in Figures 1 and 3-6. By judiciously choosing the estimator $\lambda^* = X_{(\beta)}$ as a function of the selected subset size r, we show that a performance better than the 20th order statistic detector can be obtained. One possible choice of $X_{(\beta)}$ for b=8 and d=5 is shown in table 3. The entries in the table are made keeping in mind that the scheme needs to tolerate up to 4 targets and that the false alarm swing for the clutter transition in the middle needs to be comparable, and if possible, better than the 20-OS detector. In order to evaluate the performance, we derive the expressions for the false alarm probability. A given design parameter is the false alarm probability in the homogeneous background. Given a value of P_f, the threshold t can be solved using the expressions for the joint probability of a false alarm and the selected subset size r, P(f,r), as follows:

$$P_f = \sum_{r=b}^{N} P(f,r) \tag{3.1}$$

where $P(f,r) = \Pr(X_0 > t\,X_{(\beta)},\ X_{(r)} < d\,X_{(b)} < X_{(r+1)} \mid \text{no target})$

The detection probability is computed using the same expression for P_f but with t replaced by t/(1+SNR). The derivation of the expressions for P(f,r) is straightforward, but tedious. The interested readers are referred to the technical report [24]. In order to find the false alarm swing with the clutters of different background, the same probability expressions derived for the interferes are used with INR replaced by CNR, the clutter power to noise power ratio. The performance of this selection and estimation test is shown in Figures 1 -6. The SE test can tolerate even a 5th target to a good degree and provide a detection probability of 0.85 for a large test signal strength. The upper false alarm swing for the SE test is better than that of the OS detector when the CNR is 20dB or higher. This is to be expected since a reasonable discrimination between clutters, based on a finite number of samples, is possible when the power levels are considerably different. In homogeneous background, the performances of both schemes are nearly the same. The above estimator assignment is tuned to the need that a scheme better than 20th OS detector is desired. Therefore, the estimator assignment has judiciously placed the order statistics 18 through 24 to achieve the purpose. Since the assignment has used only 18 and above, definitely the scheme cannot tolerate more than 6 targets. This shows that some knowledge of expected maximum number of interferers is required and that the SE test can provide an improved performance over the fixed OS test to a certain degree only. We also designed the SE tests with b=4 and d=10, b=4 and d=15 and compared their performances with the OS test and the SE test with b=8 and d=5. It is observed that the SE (b=8) performs slightly better than the SE (b=4). Therefore it is clear that for a specific situation, the parameters of the SE test can be fine tuned for the best performance. In order to evaluate the performance of the SE test for target models other than the Swerling type I, one has to resort to simulation studies. At the present time we have not done any such simulation studies.

In Table 4 we compare the performances of the VTM test and the SE test for the situation considered in reference [16]. The designed false alarm probability is 10^{-3}, b=8, d=5, and N=24. For large INR of the order of 20dB, the SE test exhibits much smaller false alarm increase in clutter transition. Performances of the two schemes with regard to other parameters are comparable. One reason for the slight difference in the performances is that the VTM uses (i) only the selected subset samples for the estimation and that (ii) the selection scheme makes no attempt to infer whether a clutter transition in the middle occurs. This is because the parameter k is well above N/2. In the SE method, the selection is based on $X_{(b)}$ and deliberately a low value of b below N/2 is considered. The principle of selection used in both the schemes is the same. Apart from the draw backs of (i) and (ii) for the VTM test, both the tests use a selection scheme whose performance is far from ideal. This is a contributory reason for the limited success of the SE test. It is not clear whether a selection scheme better than the one employed is possible.

IV CONCLUSION

A selection scheme widely used in selection and ranking applications is applied for the estimation of noise power level in multiple target CFAR detection. With a proper choice of the design parameters, the selection and estimation test performs better than the OS detector. Further improvement is possible if a better selection procedure can be found.

ACKNOWLEDGMENT

The authors thank Professor S. Panchapakesan for very useful discussions.

REFERENCES

1. Hansen, V.G., "Constant false alarm rate processing in search radars," in proceedings of the IEEE 1973 international radar conference, London, 1973, pp. 325-332.
2. Steenson, B.O., "Detection performance of a mean-level threshold," IEEE Transactions on Aerospace and Electronic Systems, July 1968, pp. 529-534.
3. Hansen, V.G., and Sawyers, J.H., "Detectability loss due to greatest of selection in a cell averaging CFAR," IEEE transactions on Aerospace and Electronic Systems, Jan 1980, pp. 115-118.
4. Trunk, G.V., "Range resolution of targets using automatic detectors," IEEE Transactions on Aerospace and Electronic Systems, Sept. 1978, pp. 750-755.
5. Dillard, G.M., "Mean-level detection of nonfluctuating signals," IEEE Transactions on Aerospace and Electronic Systems, Nov. 1974, pp. 795-799.
6. Rickad, J.T., and Dillard, G.M., "Adaptive detection algorithm for multiple target situations," IEEE transactions on Aerospace and Electronic Systems, July 1977, pp. 338-343.
7. Rohling, H., "Radar CFAR thresholding in clutter and multiple target situations," IEEE transactions on Aerospace and Electronic Systems, July 1983, pp. 608-621.
8. Rohling, H., "New CFAR processor based on order statistic," in proceedings of the IEEE International radar conference, Paris, 1984, pp. 38-42.
9. Finn, H.M., "A CFAR design for a widow spanning two clutter fields," IEEE Transactions on Aerospace and Electronic Systems, Mar. 1986, pp. 155-169.
10. Gandhi,P., and Kassam, S.A., "Analysis of CFAR processors in nonhomogeneous background," IEEE transactions on Aerospace and Electronic Systems, July 1988, pp. 427-445.
11. Ritcey, J.A., "Censored mean level detector analysis," IEEE transactions on Aerospace and Electronic Systems, July 1986, pp. 443-454.
12. Goldstein, G.B., "False alarm regulation in log-normal and Weibull clutter," IEEE transactions on Aerospace and Electronic Systems, Jan. 1973, pp. 84-92.
13. Weber, P., and Haykin, S., "Ordered statistic CFAR processing for two-parameter distributions with variable skewness," IEEE transactions on Aerospace and Electronic Systems, Nov. 1985, pp. 819-821.
14. Barkat, M., and Varshney, P.K., "A weighted adaptive cell averaging CFAR detector for multiple target situations," proceedings of the 21st annual conference on information sciences and systems, Johns Hopkins University, Baltimore, 1987, pp. 118-123.

15. Holm, J.R., and Ritcey, J.A., "On the Neyman-Pearson optimality of the censored mean level detector," Proceedings of the conference on information sciences and systems, Johns Hopkins University, Baltimore, 1989,pp. 346.
16. Ozgunes, I., Gandhi, P., and Kassam, S.A., "A CFAR detection scheme based on modified trimmed mean thresholding," Proceedings of the conference on information sciences and systems, Johns Hopkins University, Baltimore, 1989, pp. 356-361.
17. Weiss,M., "Analysis of some modified cell-averaging CFAR processors in multiple target situations," IEEE transactions on Aerospace and Electronic Systems, Jan 1982, pp. 102-113.
18. Hawkins, D.M., "Identification of outliers," Chapman and Hall, London, 1980.
19. Gupta, S.S., and Panchapakesan, S., "Multiple decision procedures," John Wiley & Sons, NY, 1979.
20. Gibbons, J.D., Olkin, I., and Sobel, M., "Selecting and ordering populations: a new statistical methodology," John Wiley & Sons, NY, 1977.
21. Gupta, S.S., and Huang, D.Y., "Multiple statistical decision theory: recent developments," Springer-Verlag, NY, 1981.
22. Mood, A.M., Graybill, F.A., and Boes, D.C., "Introduction to the theory of statistics," McGraw Hill, NY, 1974.
23. David, H.A., "Order statistics," John Wiley & Sons, NY, 1981.
24. R. Viswanathan and Abbas Eftekhari, "Application of selection procedure to multiple targets in clutter detection," technical report, SIUC/DEE/TR 90-8, June 1990.
25. Prashant P. Gandhi and Saleem A. Kassam, "An adaptive order statistic constant false alarm rate detector," presented at the Conference on Information Sciences and Systems, Princeton, NJ, March 1990.

Selection Probability

r	Homog. Back.	#interf.=5 INR =10dB	#interf.=5 INR = 20dB	Clutt.in middle CNR= 10dB	Clutt. in middle CNR= 20dB
8	5.e-4	7.e-4	8.e-4	0.0017	0.0035
9	0.003	0.0043	0.0047	0.0115	0.0258
10	0.0097	0.0142	0.0158	0.04	0.1
11	0.0228	0.0337	0.0378	0.0953	0.263
12	0.043	0.0638	0.0718	0.1665	0.436
13	0.0689	0.101	0.1138	0.2128	0.143
14	0.0965	0.1371	0.1535	0.2	0.024
15	0.12	0.161	0.177	0.1429	0.0025
16	0.1347	0.163	0.1725	0.0786	2.e-4
17	0.1359	0.1402	0.1371	0.0339	1.e-5
18	0.1232	0.099	0.082	0.0116	5.e-7
19	0.0994	0.0542	0.0301	0.0031	1.e-8
20	0.0703	0.021	0.0025	7.e-4	4.e-10
21	0.0422	0.0054	1.e-4	1.e-4	8.e-12
22	0.0204	9.e-4	2.e-6	1.e-5	1.e-13
23	0.0072	8.e-5	2.e-8	9.e-7	0
24	0.0014	3.e-6	1.e-10	3.e-8	0

b=8 d= 3

Table 1 Selection Probability

	Selection Probability				
r	Homog. Back.	#interf.=5 INR =10dB	#interf.=5 INR = 20dB	Clutt.in middle CNR= 10dB	Clutt. in middle CNR= 20dB
4	1.e-4	1.e-4	2.e-4	2.e-4	2.e-4
5	6.e-4	7.e-4	7.e-4	9.e-4	0.0012
6	0.0015	0.0018	0.0018	0.0026	0.0034
7	0.003	0.0036	0.0038	0.0059	0.0084
8	0.0052	0.0065	0.0068	0.0116	0.0182
9	0.0083	0.0106	0.0113	0.0211	0.0376
10	0.0123	0.0163	0.0176	0.0362	0.0771
11	0.0174	0.0239	0.0262	0.0594	0.1661
12	0.0237	0.0336	0.0374	0.093	0.4242
13	0.0311	0.0459	0.0519	0.13	0.2
14	0.0397	0.0607	0.0701	0.1553	0.0521
15	0.0494	0.0785	0.0927	0.1555	0.0093
16	0.0601	0.0986	0.1198	0.1309	0.0013
17	0.0715	0.1202	0.1508	0.0933	1.e-4
18	0.0831	0.14	0.1818	0.0565	1.e-5
19	0.0944	0.1491	0.1935	0.029	9.e-7
20	0.1041	0.1188	0.0308	0.0125	5.e-8
21	0.1107	0.0638	0.0025	0.0043	2.e-9
22	0.1113	0.022	9.e-5	0.0012	8.e-11
23	0.1011	0.0045	2.e-6	2.e-4	2.e-12
24	0.0709	4.e-4	2.e-8	2.e-5	2.e-14

b=4 d=10

Table 2 Selection Probability

r	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
β	21	21	22	24	24	21	21	20	19	19	19	19	19	18	18	18	18

Table 3 A Possible Estimator Assignment

CNR dB	False Alarm Probability SE (ES.ASS. II)	 VTM (k=20 γ = 0.75)
10	1.51e-2	1.50e-2
20	2.13e-3	1.57e-2
30	4.55e-4	1.58e-2

No. of Interferers	Detection Probability (SNR=15dB) SE (ES.ASS.II)	 VTM (k=20 γ = 0.75)
3	0.661	0.66
4	0.585	0.59

r	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
β	21	21	22	24	24	22	22	21	20	20	20	20	20	20	20	20	20

Estimator Assignment II

Table 4

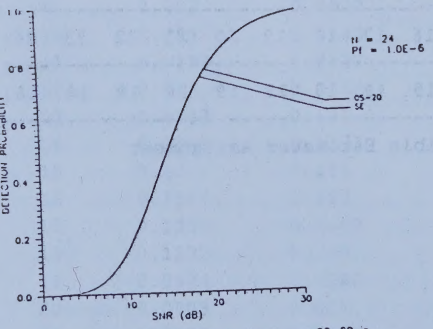


Figure 1. Probability of Detection of SE and OS-20 in Homogeneous Background.

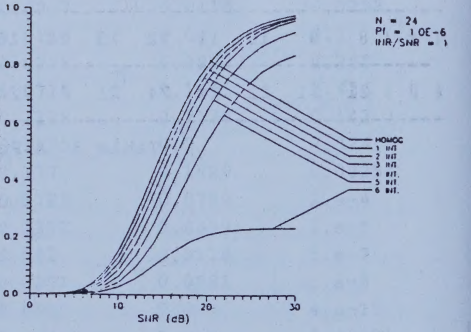


Figure 2. Probability of Detection in Multiple Targets Environment.

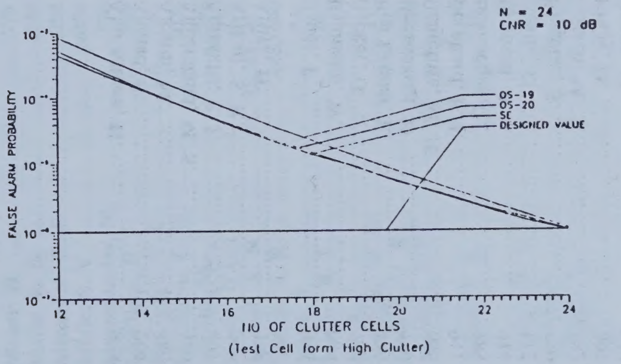


Figure 3. False Alarm Rate Performance of SE, OS-20 and OS-19 Processor in Clutter Transition (Upper Swing)

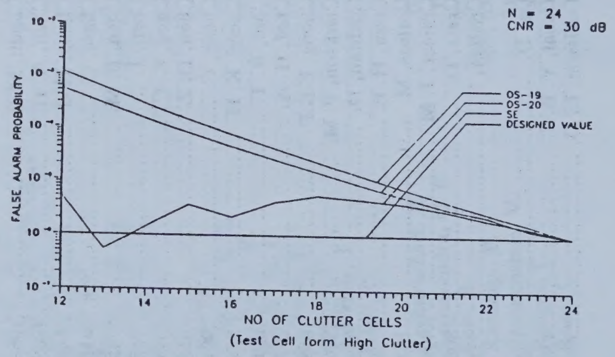


Figure 5. False Alarm Rate Performance of SE, OS-19 and OS-20 Processor in Clutter Transition (Upper Swing).

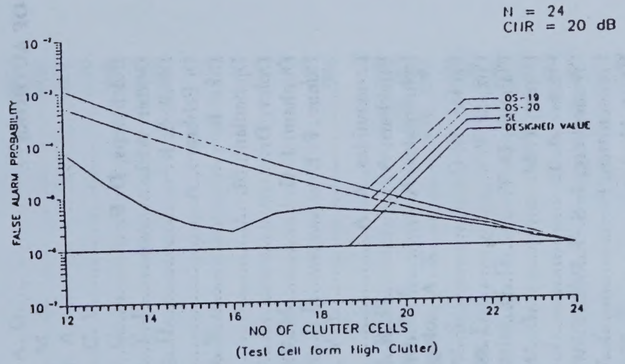


Figure 4. False Alarm Rate Performance of SE, OS-19, and OS-20 Processor in Cluttert Transition (Upper Swing).

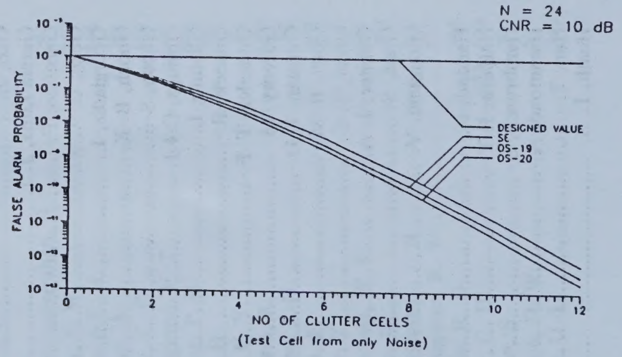


Figure 6. False Alarm Rate Performance of SE, OS-19, and OS-20 Processor in Clutter Transition (Lower Swing).

INDEX OF AUTHORS

Abu-Amara, H. 858
Ahmad, S. 192, 204
Aicardi, M. 448, 640
Andersland, M. S. 149
Antsaklis, P. J. 475, 487, 497
Arikati, S. R. 292
Attiya, H. 578

Bauer, P. 354
Baumann, W. T. 1063
Becker, G. 768
Ben Letaief, K. 429
Bensoussan, D. 779
Benteftifa, M. H. 605
Bernhard, P. J. 911
Bevilacqua, L. 783
Bhansali, A. 914
Bhatia, D. 112
Bingulac, S. 527, 529
Boesch, M. 182
Brady, D. 691
Brown, G. 548
Bui, T. N. 226
Butler, J. T. 948

Caputi, M. J. 1063
Chan, A. H. 1008
Chang, H.-J. 650
Chase, C. 455
Chen, B. M. 745
Chen, J. 755
Chen, K.-C. 398
Chen, D. Z. 818
Cheng, C. 901
Cheng, K. H. 262, 324
Chin, R. T. 601
Choi, H.-A. 272
Chong, E.K.P. 658
Clarkson, P. M. 376
Cochran, D. 362
Conn, H. E. 788
Corless, M. 162
Cramer, J. M. 408
Crass, D. 808
Cristian, F. 531

Da, D. 162
Dallal, Y. E. 438
Davisson, L. D. 398
Delchamps, D. F. 703
Dexter, A. L. 1044
Dhall, S. K. 294
Di Febbraro, A. 448
Diks, K. 978
Djebeniani, M. 362
Dolev, D. 531
Dunham, J. G. 713
Dunne, P. E. 938

Economides, A. A. 674
Eftekhari, A. 1074
Ephremides, A. 891

Fabien, B. C. 172
Fitz, M. P. 388, 408
Flaherty, D. T. 31
Forest, T. M. 599
Frazho, A. E. 344
Freudenberg, J. S. 755
Freudenstien, F. 172
Fürer, M. 274

Galatsanos, N. P. 601, 1061
Gao, Z. 497
Garcia-Luna, J. J. 901
Garofalo, F. 783
Geng, G. 1044
Georgiadis, L. 630
Ghosh, B. K. 715
Ghosh, S. 988
Gittings, C.J.J. 938
Glielmo, L. 783
Gnacek, R. 968
Gonzalez, T. F. 838
Goresky, M. 1008
Gorman, J. D. 1053
Gorti, B. M. 529
Greer, S. C. 713
Gubner, J. A. 684
Guha, S. 798
Gummadi, V. 858

Haddad, R. A. 617
Halpern, J. Y. 588
Hambrusch, S. 828
Hammoud, A. 598
Han, T. 236
Heath, L. S. 305

Hero, A. O. 1053
Herro, M. A. 1014
Hillery, A. 601
Hollot, C. V. 770
Hu, A. C. 202
Huang, Y. F. 334
Hwang, H. 617

Inouye, Y. 378

Jacquet, P. 924
Jankovic, M. 715
Jantzen, J. E. 149
Joung, Y.-J. 998

Kamel, R. E. 628
Kashyap, R. L. 60, 603
Katsaggelos, A. K. 601, 1061
Kayalar, S. 1038
Kimura, F. 599
Kinnersley, N. G. 958
Klapper, A. 1008
Ko, S.-J. 599
Konstantopoulos, P. 668
Kostic, Z. 374, 1024
Krishnan, R. 860
Krogh, B. H. 465
Kumar, S.P.R. 901
Kurz, L. 605

Lakshmivarahan, S. 294
Lau, S. W. 848
Lau, F.C.M. 848
Laub, A. J. 781
Lee, C. C. 419
Leighton, F. T. 305
Leng, P. H. 938
Li, H. 603
Li, R. Y. 51
Liano, K. 650
Liaw, J.-N. 60
Lin, F. 485
Lin, Z. 615
Lindsey, W. C. 388
Liu, R. 378
Livingston, J. N. 410
Longman, R. W. 172
Lou, R. D. 315
Lu, C. H. 615, 628
Luo, S. 204
Luse, D. W. 527
Ly, U.-L. 745

Maher, H. 779
Maier, R. S. 139
Manacher, G. K. 292
Maric, S. V. 1024
Mavronicolas, M. 578
McEliece, R. J. 869
McGeoch, C. C. 62
McGillem, C. D. 1026
Merritt, M. 548
Meyer, G.G.L. 733
Michel, A. N. 11, 31, 41, 487
Minciardi, R. 448, 640
Misra, J. 541
Moose, R. L. 1063
Mrad, F. 192
Mulligan, M. G. 410
Murthy, G. R. 364
Myoupo, J. F. 254

Narahari, B. 272

O'Dea, R. J. 419
Olariu, S. 282, 936
O'Rourke, J. 788
Özgüner, Ü 182

Packard, A. 765, 768, 781
Padmanabhan, P. 770
Pandey, P. 781
Panwar, S. S. 132
Papadakis, Y.N.M. 1028
Papannareddy, R. 429
Park, B. P. 507
Park, J. 21
Passino, K. M. 475, 487
Patrikar, A. 1
Pelc, A. 978
Pesenti, R. 640
Peterson, G. L. 568
Philips, T. K. 132
Pinotti, M. C. 926
Podrazik, L. J. 733
Prabhu, U. 226
Provence, J. 1
Pucci, G. 926

Raghavachari, B. 274
Raghavendra, C. S. 216
Rais, B. 924
Ramadge, P. J. 455, 658
Ramakrishna, V. 743
Rangan, C. P. 292
Rhee, C. 294

Richards, D. S. 82
Rosen, Y. 968
Rosenberg, A. L. 305
Rosenkrantz, D. J. 911

Saberi, A. 745
Sadowsky, J. S. 429
Sain, M. K. 517
Salehi, M. 701
Salowe, J. S. 82
Sandberg, I. W. 21
Santoro, N. 558
Sarrafzadeh, M. 315
Savage, C. D. 72
Schrader, C. B. 517
Schueller, K. A. 948
Schulz, T. J. 598
Schwing, J. L. 282
Sezan, M. I. 374
Shamai (Shitz), S. 438
Sherlekar, D. D. 102
Sherman, P. J. 344
Si, J. 41
Silvester, J. A. 674, 860
Sinha, A.S.C. 1038
Sivarajan, K. N. 869
Skiena, S. S. 914
Smolka, S. A. 998
Snyder, D. L. 598
Soon, V. C. 334
Sreenivas, R. S. 465
Srivastava, P. 1072
Stanat, D. F. 236
Strong, H. R. 531
Suhartanto, C. H. 1026
Suzuki, I. 588, 808
Szpankowski, W. 92, 924

Tassiulas, L. 891
Telang, V. 1014
Teng, J. 765
Thomopoulos, S.C.A. 881, 1028
Tien, S.-B. 216
Titlebaum, E. L. 374, 1024
Tong, L. 378
Tschiesche, T. L. 206
Tu, H.-Y. 828
Tyagi, A. 122
Tygar, J. D. 62

VanLandingham, H. F. 529
Verdú, S. 691
Verriest, E. I. 507
Viniotis, I. 630
Viswanathan, R. 1074

Wan, F. 296
Wang, Q. 262, 324
Wang, T.M.Y. 72
Wann, C. D. 881
Wen, Z. 936
Willett, P. J. 1072
Williamson, G. A. 376
Woo, P.-Y. 202

Xu, J. X. 182
Xu, M. 51

Yamashita, M. 808
Yaz, E. 725
Yegani, P. 159
Yen, G. 11
Yurtseven, H. O. 1038

Zazanis, M. 668
Zhang, L. 881
Zhang, J.-Y. 282
Zheng, S.-Q. 246

[illegible], D. S. 92
[illegible] 562
[illegible], A. L. 968
[illegible] 914

Saberi, A. 785
[illegible] 1072
Seki, M. 517
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]

[illegible]
[illegible]
[illegible]
Thompson, [illegible] 1028
Tsai, S.-D. 38
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]

[illegible]
[illegible]
[illegible]

[illegible] 670
[illegible], R. 1034

Wan, P. 795
Wang, G. ....[illegible]
Wang, T.-J. 75
Wang, C. D. ....[illegible]
[illegible]
[illegible]
Williamson, G. A. 376
Wu, P.-Y. ....[illegible]

Xu, J. K. 1022
Xu, M. ....[illegible]

[illegible]
[illegible]
[illegible]
[illegible]
[illegible]

[illegible], M. 543
Zhang, L. ....[illegible]
Zhang, L.-Y. ....[illegible]
Zhang, S.-[illegible] ....[illegible]

UNIVERSITY OF ILLINOIS-URBANA

621.3811AL5 C001
PROCEEDINGS / ALLERTON CONFERENCE ON COM
1990

3 0112 008149277